AFRL-RH-WP-TR-2011-0065

SYNTHESIZING THE DYNAMICS OF CYBERSPACE VISUAL RENDERINGS FOR DISBTRIBUTIVE SYSTEMS TO CHARACTERIZE CYBER ATTACK, PERFORMANCE, AND VULNERABILITY

Rodney G. Roberts, Ph.D.
Balaram Nutakki
Daniel W. Repperger, Ph.D.

Florida A&M University
Division of Research

MARCH 2011
Final Report

AIR FORCE RESEARCH LABORATORY
711 HUMAN PERFORMANCE WING,
HUMAN EFFECTIVENESS DIRECTORATE,
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

7/18/11 AM

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

Qualified requestors may obtain copies of this report from the Defense Technical Information Center (DTIC).

AFRL-RH-WP-TR-2011-0065 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

Judson L. Shattuck
Work Unit Manager
Battlespace Visualization Branch

Jeffrey L. Craig
Chief, Battlespace Visualization Branch
Warfighter Interface Division

Michael A. Stropki
Chief, Warfighter Interface Division
Human Effectiveness Directorate
711 Human Performance Wing

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
31-01-2011	Final	08 Sept 2008 - 31 Jan 2011

4. TITLE AND SUBTITLE

Synthesizing the Dynamics of Cyberspace Visual Renderings for Distributive Systems to Characterize Cyber Attack, Performance and Vulnerability

5a. CONTRACT NUMBER
FA 8650-08-C-6926

5b. GRANT NUMBER
NA

5c. PROGRAM ELEMENT NUMBER
62202F

6. AUTHOR(S)

Rodney Roberts
Balaram Nutakki
Daniel W. Repperger

5d. PROJECT NUMBER
7184

5e. TASK NUMBER
11

5f. WORK UNIT NUMBER
718411AM

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Florida A&M University, Division of Research
1500 Wahnish Way
Tallahassee FL 32307-3100

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Materiel Command
Air Force Research Laboratory
711 Human Performance Wing
Warfighter Interface Division
Battlespace Visualization Branch
Wright-Patterson AFB OH-45433-7022

10. SPONSOR/MONITOR'S ACRONYM(S)
711 HPW/RHCV

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-RH-WP-TR-2011-0065

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution A. Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

88ABW Cleared 06/07/2011; 88ABW-2011-3265.

14. ABSTRACT

The objective is to better understand and synthesize the dynamic response of complex and distributed networks when they are under cyber attack. Modern complex and distributed systems now are highly vulnerable and can be easily compromised by attacks at key nodes. Finding the key nodes or links in a model of a distributed network system is critically important to understand how to mitigate an attack. An analysis tool needs to be developed to first determine what areas of a complex network may be highly vulnerable so that commanders and other decision makers can help protect their assets. Most modern complex and distributed networks have been built in an ad hoc manner where nodes, links, and other connections spring up without any real planning. Decision makers then have to assign authority and responsibility to the key constituent players and they need a framework and basis to improve their allocation of scarce resources to protect complex networks. From another point of view, some networks are hostile and the overall objective would be to attack and take such a network down. The expression taking a network down is assumed to be equivalent to producing congestion and restricting the flow of important military assets, water, food, medicine and other important entities.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	SAR	62	Judson L Shattuck
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER *(Include area code)* (937)255-8742

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Abstract

An increasing number of Air Force related systems are being modeled as networks. This has motivated philosophical discussions concerning distributed networked operations and network centric warfare. However, successful application of such approaches will require a solid mathematical foundation. In this work, a mathematical framework is proposed that uses graph theory and information theory. Once a mathematical model of the network is established and suitable objective functions are identified to quantify the performance of the network, optimization theory is used to identify the vulnerabilities of the network. Knowledge of these vulnerabilities can be exploited to take down a network or make the network less efficient, for example through increased congestion, if that is the goal. On the other hand, if the goal is to protect the network, the proposed approach can help system designers to improve reliability. Simple examples such as a logistics distributed network for Air Terminal Operations Flight (ATOF) are used to illustrate the approach. Using a genetic algorithm, it is shown that the difference between the maximum and minimum information flows of a network can be quite significant. Identifying the maximum and minimum flows provides decision makers with an understanding of how to protect or attack a network system.

Table of Contents

Section Page

List of Figures

Figure Page

List of Tables

Table Page

Acknowledgments

The PI would like to express his sincerest appreciation to Dr. Daniel W. Repperger for his support of this work. Dr. Repperger‘s significant contributions to the use of network theory as a basis for understanding cyber warfare formed the foundation for this project. Unfortunately, Dr. Repperger did not live to see the completion of the project. This work is dedicated to his memory. The authors would also like to thank Mr. Judson Shattuck for his work taking over as the Program Manager.

Executive Summary

The ability to efficiently model and evaluate complex networks such as those found in modern military systems are critical for decision making. A combination of multiple scientific disciplines will be required to understand how highly connected systems operate. Such techniques will allow decision makers to uncover explanations for how simple processes at the level of individual nodes and links can have complex effects that ripple through the overall network. This report describes an approach to study networks based on graph theory and information theory. A graph is a mathematical structure that consists of a set of vertices and set of edges, where each edge links a pair of vertices. Components of the network are identified with the vertices, also called the nodes, and the relationship between the nodes is given by the edges. Properly formulated graphs identify patterns of interconnections among a set of elements. However, successful decision making requires more than representing the network in question. There must also be a means of quantifying the effects that individual elements and groups of elements of the network have on each other and on the overall system. In this work, the problem of quantifying this effectiveness is addressed using information theory. Originally developed to solve fundamental problems in the theory of communication systems, information theory also has applications in economics, computer science, probability theory, statistics, and a number of other important areas. The next problem addressed in the work is the network flow problem. Maximizing and minimizing the network flow allows the decision maker to identify the effectiveness of the network and to identify critical flows that should be protected or attacked, depending on the goals of the decision maker. Unfortunately, optimizing the network flow is generally a difficult problem. To address this problem, genetic algorithms are used.

The concepts developed in this work are carefully illustrated with simple examples. To demonstrate the efficacy of the approach, a specific Air Force related logistic distributed network system is studied in Section 6. This problem is formulated as an optimization problem in Section 7 and a discussion of the results appears in Section 8.

In conclusion, a systematic approach has been developed to study the efficiency of networks. This approach uses graph theory to determine the structure of the network and genetic algorithms to study the minimum and maximum flow of the network. The ability to study and optimize network flow gives decision makers the ability to identify critical and vulnerable nodes and the opportunity to induce or mitigate congestion. The efficacy of the approach was illustrated with some simple examples. Further recommendations are that more difficult and realistic Air Force related problems should be studied using the developed methods.

1.0 INTRODUCTION

In [1] J. Cares has identified some current challenges within the study of network centric systems to characterize military and other complex network interactions. However, his work stressed very little on the mathematical aspect of such systems. Obviously characterizing any system mathematically quantifies many things. This shows the need for developing some basic mathematical techniques to analyze complex networks. The approach followed in this work is to synthesize mathematical procedures to quantify performance in terms of information or other flow variables that may occur in those networks. A general assumption is that performance is proportional to the high rate of flow of information through systems. For human machine systems, the flow arrows indicate the input and output of several human-machine systems. These inputs and outputs may consist of different units such as telephone calls, emails, mouse clicks, etc. A paradigm of resources working on a common task in a team environment can be used to describe these human machine interactions. Performance can be assumed to be proportional to a sufficient throughput while traversing the network. A very general assumption is that a network with a low level of information flow, i.e., a congested network, may not have the best of performance. However, when we take reliability and reduced vulnerability (less likely to be attacked), the congested networks may provide alternative advantages in being resistant to some types of attack. This is much like the trade-off between speed and accuracy. In certain tasks speed may be more important than the accuracy and for some systems accuracy might be of more interest. Thus high or low levels of information flow may be interpreted depending upon the type of task performed.

In this report, we describe a mathematical framework for network centric systems that is applicable to military type systems. Such a framework can serve as a basis for studying important network issues such as cyber attacks. The next section describes graph theory concepts that form the basis for the proposed approach.

2.0 INTRODUCTION TO GRAPH THEORY

Basic procedures from graph theory, which will enable the generalization of this methodology to complex networks, are discussed in this section.

2.1 Basic Graph Theory Notation

A directed graph *G=(N, A)* consists of a set *N* of nodes which are also referred as vertices and a set *A* of arcs which are also called as links or edges. The elements in an arc are ordered pairs of distinct nodes. Figure 1 gives an example of a directed graph. For this graph, *N={1, 2, 3, 4, 5, 6, 7}* and *A = {(1,3), (2,1), (2,3), (2,4), (3,5),(3,6),(4,7),(4,5),(5,2),(6,7),(7,5)}*. A directed network is a directed graph whose nodes or arcs have associated numerical values.

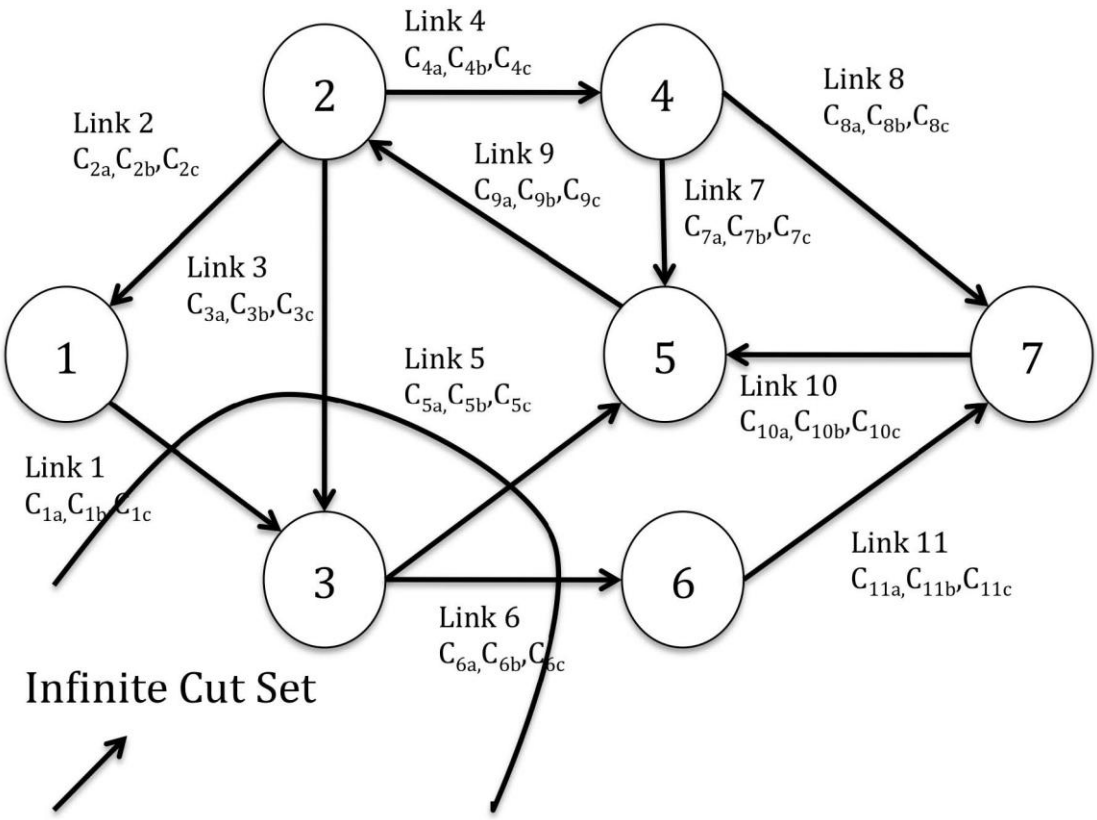


Figure 1. Network with Cut Set and Flow Capacity Measures

The characteristics of link *i* is given by the triplet *[c_{ia}, c_{ib}, c_{ic}]* where *i* = link number, c_{ia} = the capacity of the link, c_{ib} = the actual flow (thus $|c_{ib}| \leq c_{ia}$), and c_{ic} is the cost per unit flow to traverse this arc. One useful tool from graph theory to characterize the structure of a network is the adjacency matrix.

2.2 Node-Node Adjacency Matrix

An adjacency matrix describes the network as an $n \times n$ matrix, *H={h_{ij}}*, where *n* is the number of nodes present in the network. This matrix has rows and columns corresponding to every node and its ij^{th} entry h_{ij} equals 1 if *(i,j)* $\in A$ and equals 0 otherwise [2]. Figure 2 specifies this representation for the network shown in Figure 1. Since link 1 starts at node 1 and ends at node 3, a "1" appears in first row and third

column of the *H* matrix. The number of non-zero elements of the adjacency matrix is equal to the number of links in the network. The Frobenius norm of the adjacency matrix provides a measure for connectivity of a complex network. Recall that the Frobenius norm of a matrix is the square root of the sum of the squares of the elements in the matrix. A larger value of the Frobenius norm implies high interconnectivity within the network.

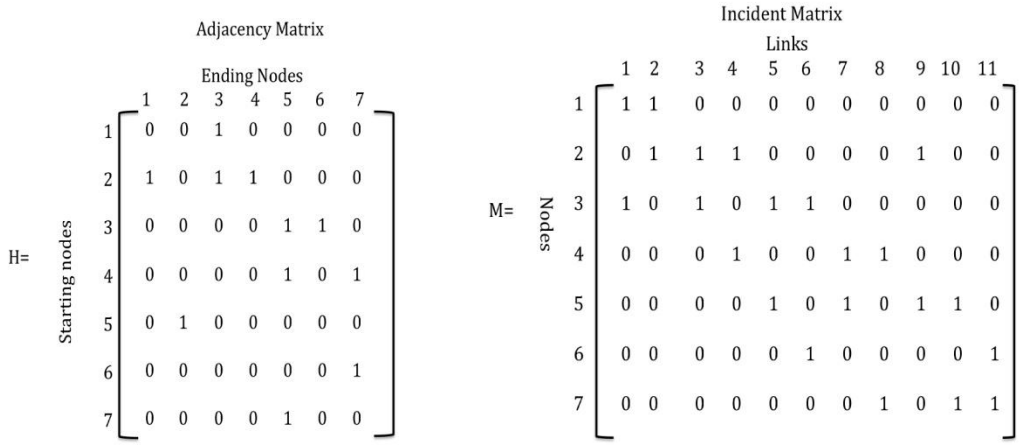


Figure 2. Adjacency and Incident Matrices for Figure 1

Figure 2 also displays the incident matrix of the network shown in Figure 1. The nodes correspond to the rows, and the links correspond to the columns in this matrix. Since link 1 starts at node 1 and ends at node 3, a ―1" appears at both first and third rows of first column. Each column in an incident matrix has exactly two ones.

In Figure 1, a ―cut-set" is specified by a curve that divides the network into two or more parts. A network may have a source vertex (denoted by ‗s') that generates the flow and a sink vertex (denoted by ‗t') that absorbs the flow. If a cut-set does not isolate a source or sink, then at a node, such as vertex 3, the net sum of the flows should be zero (Kirchoff's law) through the cut set [3] and denoted via:

$$c_{1b} + c_{3b} - c_{5b} - c_{6b} = 0 \quad (1)$$

where the flows entering the node are considered positive and the flows leaving a node are negative. If, however, a cut set isolates a source or sink, the total flow impacting the cut set would be the value of the flow out of a source or into the sink.

2.3 Maximum Flow Problem

The maximum flow problem is to find the maximum flow possible between two special nodes, a source node s and a sink node t, without exceeding the capacity of the edge. Consider a capacitated network $G = (N, A)$ with ‗s', ‗t' ∈ N being the source and sink of G, respectively. The capacity of an edge is a mapping $c : A \to R^+$, denoted by c_{uv} that represents the maximum amount of flow that can pass through this edge. A flow is a mapping $f : A \to R^+$, denoted by f_{uv}, subject to the following two constraints:

1.) $f_{uv} \leq c_{uv}$ for each (u,v) ∈ A (capacity constraint: the flow of an edge cannot exceed its capacity)

2.) $\sum_{u:(u,v)\in A} f_{uv} = \sum_{u:(v,u)\in A} f_{vu}$, for each $v \in V \setminus \{s,t\}$ (conservation of flows: the sum of the flows entering a node must equal the sum of the flows exiting a node, except for the source and the sink nodes).

The value of flow is defined by $|f| = \sum_{v\in V} f_{sv}$ where ‗s' is the source of A. It represents the amount of

flow passing from the source to the sink. The maximum flow is to maximize $|f|$, that is, to route as much flow as possible from ‗s' to ‗t'. The maximum value of an s-t flow is equal to the minimum capacity of an s-t cut in the network, as stated in the max-flow min-cut theorem [4]. This problem can be formulated as

$$\text{Maximize the flow} = \sum_{u} (f_{su} - f_{us}) \tag{2}$$

$$\text{or Minimize the quantity: (cost * flow)} = \sum_{u,v} (C_{su} * f_{su} - C_{sv} * f_{sv}) \tag{3}$$

where C_{su} is the cost associated with that link. The above equations are subjected to the constraints stated above.

Flow optimization of a complex network can be difficult using conventional optimization techniques. One approach is the use of genetic algorithms. The next section is a brief discussion on genetic algorithms.

3.0 GENETIC ALGORITHM

As early as the 1950‘s, the biological metaphor of evolution was being applied to computation. As computational power has increased and the foundations of these evolutionary algorithms have been formalized and improved, evolution based approaches have increasingly been used in the solution of optimization problems. Currently, Bäck [5] identifies three strongly related but independently developed approaches to evolutionary computation: genetic algorithms, evolutionary programming, and evolution strategies. Although identified as distinct aspects of evolutionary programming, the three evolutionary strategies are closely related by their mimicry of natural evolutionary processes. The work described here is concerned with the application of genetic algorithms as optimization tools.

Since their formal introduction in 1975 by Holland [6], genetic algorithms have been applied to a variety of fields—from medicine and engineering to business—to optimize functions which do not lend themselves to optimization by traditional methods, and other applications of genetic algorithms include automatic programming and simulation of natural systems. More recently, the study and practical development of genetic algorithms by Goldberg and DeJong [7,8] has resulted in great growth in the application of genetic algorithms to optimization problems. As succinctly stated by Goldberg, genetic algorithms are ―search procedures based on the mechanics of natural selection and natural genetics.” Random choice is used as a tool to guide a global search in the space of potential solutions.

Genetic algorithms differ from traditional optimization and search methods in several respects. Rather than focusing on a single candidate solution (point in design space), genetic algorithms operate on populations of candidate solutions, and the search process favors the reproduction of individuals with better fitness values than those of previous generations (optimal individuals). Whereas calculus-based and gradient (hill climbing) methods of solution are local in the scope of their search and depend on well-defined gradients in the search space, genetic algorithms are useful for dealing with many practical problems containing noisy or discontinuous fitness values. Enumerative searches are also inappropriate for many practical problems. Because they exhaustively examine the entire search space for solutions, they are only efficient for small search spaces, while the global scope of the genetic algorithms makes them suitable for problems with large search spaces. Thus, genetic algorithms not only differ in approach from traditional optimization methods but they also offer an alternative method for cases in which traditional methods are inappropriate [9].

Genetic algorithms have been applied (or misapplied) to continuous optimization problems, but that is rarely as effective as continuous optimization methods. Evolutionary programming is appropriate for continuous problems, while genetic algorithms, being inherently discrete, are not. The genetic algorithm as a discrete optimization process is distinct from more conventional optimization techniques in five ways:

1. Genetic algorithms encode designs (feasible points) in a string, and it is this encoding which the genetic algorithm works with: each individual in a population is an encoding of a possible solution to the discrete optimization problem being analyzed.

2. Genetic algorithms work simultaneously with a population of designs, not a single design or candidate solution.

3. Genetic algorithms use only an objective function to evaluate candidate solutions, not derivatives or other auxiliary information.

4. Genetic algorithms use random change in their search, not (solely) deterministic rules. The process used by genetic algorithms to evolve solutions to optimization problems is analogous to

the natural process of evolution by natural selection. Evolution as a natural process allows complex, highly adapted organisms to develop and thrive in an environment through the processes of genetic change and natural selection. Sexual reproduction (sexual in the sense of occurring between two parent individuals as opposed to one) provides for the preservation of existing genetic information and the creation of new genetic information, and individuals in a population survive based on their fitness in their environment. Fitness is a quality measure of an individual‘s viability with respect to such criteria in the natural environment as food supply, competition for food and mates, and predation. The genetic information carried by more fit individuals is more likely to be passed on to ensuing generations simply because more fit individuals are more likely to survive to reproduce—Darwinian survival of the fittest.

5. Genetic Algorithms apply the natural evolutionary processes of evaluation and selection to string representations of the arguments of the function being optimized. Structures (individuals in natural systems) are encoded into one or more strings (chromosomes). These individuals reproduce, and fit individuals persist from generation to generation, yielding improved designs.

The structure is analogous to the phenotype in natural systems and corresponds to a candidate solution to the optimization problem or a point in the design space, while the string encoding of the arguments to the function being optimized is analogous to the genotype. A decoding from the string representation to the structure is made for the purpose of fitness analysis by the objective function. The objective function yields a quantitative measure of an individual‘s utility or goodness, to be used as a selection criterion.

For sets of individuals (population), evolution is simulated by means of reproduction and random genetic changes affected by genetic operators, and survival of the fittest is accomplished by first evaluating each structure‘s objective function value, and then selecting for reproduction and survival those structures which fit a predetermined selection criterion, biased towards selecting fitter individuals. The search is exploitative: selection is accomplished by analyzing the objective function value with the goal of preserving genetic information, which minimizes the objective function. More formally, the aim of the search is to identify an approximation of the global minimum of a real-valued objective function f: M → E, by evolving a solution $x_* \in M$ such that $f(x) \geq f(x_*)$ for all $x \in M$. The process is analogous if the goal is to maximize the objective function.

3.1 Designing a Genetic Algorithm

The genetic algorithm is a heuristic search process, and its behavior is governed by the following design choices:

1. How shall an individual be represented—what are the values taken on by the genes in the chromosome strings encoding the arguments to the function being optimized?

2. Creation of individual and mechanism of reproduction.

3. Genetic operators used in accomplishing the genetic processes.

4. By what means will fit individuals in a population be selected for reproduction—what is the mechanism of selection?

The following sections discuss these choices in more detail.

3.2 Representation of Individuals

The genetic information that is operated on by a genetic algorithm is contained in the chromosomes. A chromosome is a finite length string and is made of genes. A gene is an encoded variable of the problem being optimized and it will take values from alphabets. An alphabet is analogous to a set of alleles in nature. The nature of the alphabet used for encoding depends upon the problem being optimized. Generally cardinality also depends upon the problem. According to Goldberg‘s principle of meaningful building blocks and principle of minimum alphabets [7], low cardinality alphabets are recommended and often binary or gray codes are used for encoding. High cardinal alphabets are used if the problem requires them.

3.3 Reproduction of Individuals

In genetic algorithms, evolution from generation to generation is simulated both by preserving the genetic information contained in the chromosome strings of fit individuals and by altering this information by means of random genetic changes. Genetic operators affect both of these goals.

Crossover is a genetic operator that combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to a user-definable crossover probability. The probability for chromosome crossover is around 60 - 70%.

During one point crossover, a crossover operator that randomly selects a crossover point within a chromosome then interchanges the two parent chromosomes at this point to produce two new offspring. Consider the following two parents, which have been selected for crossover. The ”|” symbol indicates the randomly chosen crossover point.

Parent 1: 11001|010
Parent 2: 00100|111

After interchanging the parent chromosomes at the crossover point, the following offspring are produced:

Offspring1: 11001|111
Offspring2: 00100|010

During two-point crossover, a crossover operator that randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offspring. Consider the following two parents which have been selected for crossover. The ”|” symbols indicate the randomly chosen crossover points.

Parent 1: 110|010|10
Parent 2: 001|001|11

After interchanging the parent chromosomes between the crossover points, the following offspring are produced:

Offspring1: 110|001|10
Offspring2: 001|010|11

3.4 Introducing Random Genetic Change

The goal of introducing change to the information in the chromosome strings of individuals created by crossover is achieved with the mutation, addition, deletion, and permutation operators. The mutation operator alters the information of a chromosome by changing the value of a gene present in that

chromosome. This is achieved by generating a random variable and comparing it with mutation rate. Mutation rate is the chance that a particular gene in the chromosome is flipped. This is usually a very low value for a binary encoded gene, say 0.01.The following example illustrates the mutation operator. A randomly determined gene in the chromosome is changed. The mutated gene is indicated as a bold character.
Before Mutation:
[111 110 101 010 101 010]

After Mutation:
[111 110 101 010 **0**01 010].

The addition operator randomly adds a gene to the chromosome string. In this, a randomly determined gene from offspring 2 is added at a random point in offspring1. The randomly selected addition point is denoted by the symbol | . In this example, addition causes the number of actual genes in the chromosome to increase from 15 to 16.

Chromosome before addition
[3 2 3 1 3 3 2 1 3|2 3 1 0 0 0]

Chromosome after addition
[3 2 3 1 3 3 2 1 3 3 2 3 1 0 0 0].

The deletion operator is similar to addition except that instead of adding a new gene an existing gene present at a randomly generated point is deleted from the chromosome. In this example, deletion causes the number of actual genes in the chromosome to decrease from 15 to 14.

Chromosome before deletion
[3 2 3 1 3 3 2 1 3 3 2 3 1 0 0]

Chromosome after deletion
[3 2 3 1 3 2 1 3 3 2 3 1 0 0].

The permutation operator relays information from one part of the chromosome to another by inverting the order of a randomly determined sequence of genes. In the following example, the points at which the permutation operator is applied are indicated by the symbol | .

Chromosome before permutation
[3 2 3|1 3 2 1 3 3|2 3 1 0 0 0]

Chromosome after permutation
[3 2 3|3 3 1 2 3 1|2 3 1 0 0 0].

3.5 Selection for Reproduction

In genetic algorithms, the goal of simulating natural selection is achieved by implementing a selection mechanism. Every individual chromosome is encoded as a string. In every generation this chromosome string is decoded and its fitness value, which is a measure of its quality, is calculated using an objective function. Individuals are selected randomly from the population for mating where the individuals with higher fitness values are given higher priority.

Biasing of the selection criteria is accomplished using roulette wheel selection. Roulette selection is

calculated on the normalized fitness values of a population. Normalization is the summing of the fitness for the entire population and then dividing each individual‘s fitness by this sum. The probability of a particular chromosome being selected depends on its normalized value, i.e., the higher the normalized value, the higher the probability that chromosome will be selected. A uniform random variable between 0 and 1 is generated; the chromosome with the normalized fitness value close to this variable is selected for mating.

Using a selection criterion instead of simply selecting chromosomes with the highest fitness value ensures that the individual with the highest fitness value does not dominate the succeeding generation. In order to preserve the properties of the best chromosome in the current population, it is added to the child population after the child with least fitness value is discarded. This is known as *elitist selection*. There are several other methods other than roulette selection like tournament selection where two individuals are randomly selected from the current population and the individual with the highest fitness value between them is selected for mating.

3.6 The Genetic Algorithm

Finally, the Genetic Algorithm (GA) may be expressed algorithmically. The implementation of this algorithm is referred to as the standard GA (sGA) [3]:

```
gen=0, initialize population with a default structure (chromosome).
Calculate the fitness value for each of the structure.
While not terminated
        Do
        Generate child structures (offspring) from population by crossover and then apply genetic
            operators (mutation) to the children.
        Calculate the fitness value of the children.
        If the fitness value is greater than the least value then update the population with this child.
        Select from updated population (Using roulette method).
        gen=gen+1.
End.
```

This can be terminated after a certain number of generations after which there is no real change in the fitness value or after the fitness value has reached a satisfactory value. Here gen is an integer value, which represents the number of the generation. Population is referred as a set of structures (chromosomes).

4.0 INFORMATION THEORY

In this section we introduce most of the basic definitions required for the subsequent development of the theory. After defining entropy and mutual information we establish the relationship between them and prove the non-negativity of the mutual information. For any probability distribution, we define a quantity called entropy, which has many properties that agree with the intuitive notion of what a measure of information should be. This notion is extended to define mutual information, which is a measure of the amount of information one random variable contains about another. Mutual information is a special case of a more general quantity called relative entropy, which is a measure of the distance between two probability distributions [10]. All these quantities are closely related and share a number of simple properties, some of which are discussed in this section.

4.1 Entropy

Entropy is a measure of the uncertainty in a random variable. If *p(x)* is the probability of the event $X=x$ where is X is a discrete random variable then we can define entropy as

$$\mathrm{H(X)} = \sum_{x \in X} p(x) \log\left[\frac{1}{p(x)}\right]. \tag{4}$$

The log used here is a logarithm of base 2. Generally entropy is expressed in bits. If the logarithm used is of base other than 2 then it is denoted as $H_b(x)$ and is expressed in nats [10].

We will use the standard convention that $0 \cdot \log(0) = 0$. The fact that this convention is well defined follows from the observation that $\lim_{x \to 0} x \log(x) = 0$.

4.1.1 Joint Entropy

Joint entropy characterizes the uncertainty in two random variables. The joint entropy *H(X, Y)* of a pair of discrete variables (X, Y) with a joint distribution p(x, y) is defined as

$$\mathrm{H(X,Y)} = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log\left[\frac{1}{p(x,y)}\right]. \tag{5}$$

4.1.2 Properties of Entropy

1. Entropy is always greater than or equal to zero. Entropy is equal to zero when the probability of the event is zero

$$H(X) \geq 0. \tag{6}$$

2. Let $P=\{p(x_1), p(x_2), \ldots.\}$ be the probability distribution of X, then H(X) is continuous in P.

3. If X and Y are jointly distributed random variables, then $H(X,Y) \leq H(X)+H(Y)$ with equality if and only if X and Y are statistically independent.

4. Entropy is a symmetric function:

$$H(X,Y)=H(Y,X). \tag{7}$$

4.2 Conditional Entropy

Conditional Entropy is the remaining uncertainty (entropy) in a random variable given that the value of the other random variable is known. It is referred to as entropy of ‗X' conditional on ‗Y' and is written as *H(X|Y)*. Conditional entropy *H(X|Y)* is the weighted average (with respect to the probabilities *p(y)*) of the entropies *H(X|y)* over all the possible values of *Y*. It is computed as

$$\mathrm{H(X\,|\,Y)} = \sum_{x\in X}\sum_{y\in Y} p(y)\,p(x\,|\,y)\cdot\log\left(\frac{1}{p(x\,|\,y)}\right) \tag{8}$$

where *p(x|y)* is the probability that *X=x* given that *Y=y*.

4.2.1 Chain Rule for Joint Entropy

The joint entropy of a pair of random variables is equal to the sum of entropy of the one and the conditional entropy of the other:

$$H(X,Y)=H(X)+H(Y|X). \tag{9}$$

It should be noted that *H(X|Y)* is not equal to *H(Y|X)*. However, *H(X,Y)* is equal to *H(Y,X)*.

4.3 Mutual Information

Entropy in its basic form is a measure of uncertainty rather than a measure of information. Specifically, the entropy of a random variable gives the amount of information needed to describe that random variable. When the entropy of a random variable is large that means more information is needed to describe that random variable. Although conditional entropy can tell us when two variables are completely independent, it is not an adequate measure of dependence. So we choose mutual information as a measure, which gives the amount of information that a variable contains about the other one. Mutual information *I(X;Y)* measures how much the realization of random variable *Y* tells us about the realization of *X*; simply it is a measure of reduction of randomness (uncertainty) of a variable given knowledge of another variable:

$$I(X;Y)=H(X)-H(X|Y). \tag{10}$$

We can express equation (9) in one additional way by invoking the chain rule

$$I(X;Y)=H(X)+H(Y)-H(X,Y). \tag{11}$$

This equation can be described within the context of a Venn diagram below [11]

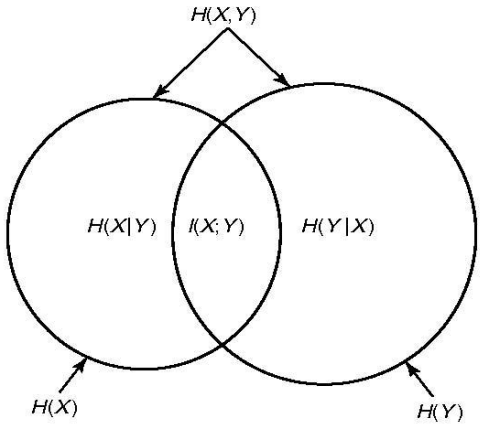


Figure 3. Information Variables as a Venn Diagram

4.3.1 Properties of Mutual Information

1. Mutual information of a group of random variables is always a non-negative number
$$I(X;Y) \geq 0. \quad (12)$$

2. Mutual information satisfies the symmetric property
$$I(X;Y)=I(Y;X). \quad (13)$$

3. Mutual information between a random variable and itself is simply its entropy
$$I(X;X)=H(X). \quad (14)$$

Mutual information does not satisfy the triangular inequality since $I(X;Z) \leq I(X;Y)+I(Y;Z)$ is not always true for three random variables X,Y,Z. For this reason we cannot consider mutual information as a metric, so we introduce two new terms: the relative information distance metric D_R and the efficiency measure E_f.

4.4 Relative Information Distance and Efficiency Measure

From Figure 4 we can say that the information channel consists of five variables as originally defined by Shannon, in which only three are independent [10]. As illustrated in the figure, the information channel is independently characterized by the three variables on the right hand side. We can define the relative information distance *metric* D_R as
$$D_R = H(X|Y)+H(Y|X). \quad (15)$$
From the definitions of conditional entropies we can express D_R as
$$D_R = 2H(X,Y)-H(X)-H(Y)=H(X)+H(Y)-2I(X;Y).$$
The efficiency measure can be defined as
$$E_f = I(X;Y)/H(X) \quad for\ H(X)>0. \quad (16)$$

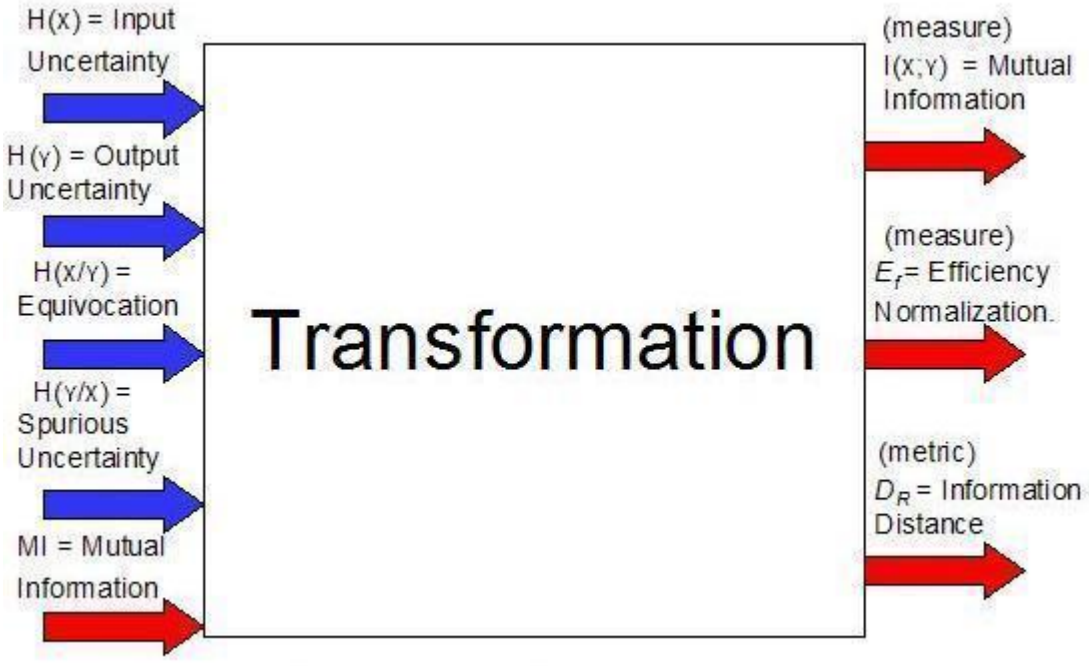


Figure 4. Transforming the Communications Channel into Information Variables

All five Shannon variables on the left hand side can be written in terms of their three key constituent information quantities (I, E_f, D_R) and vice versa. The following eight relationships show the unique objective mapping that exists between the five uncertainty variables derived by Shannon and three parsimonious variables D_R, E_f, and $I(X;Y)$ [3]:

$$H(X) = [I(X;Y)] / E_f, \text{ for } E_f > 0 \quad (17)$$

$$H(X/Y) = [I(X;Y) (1- E_f)] / E_f \quad (18)$$

$$H(Y/X) = D_R – I(X;Y) (1 – E_f)] / E_f \quad (19)$$

$$H(Y) = I + D_R – I(X;Y) (1 – E_f) / E_f \ . \quad (20)$$

$$I(X;Y) = I(X;Y) \text{ (this variable was originally an information variable)} \quad (21)$$

Conversely, D_R, E_F and I on the right side of Figure 6 satisfy:

$$D_R = H(X|Y) + H(Y|X) \quad (22)$$

$$E_f = [I(X;Y)] / H(X), \text{ for } H(X) > 0 \quad (23)$$

$$I(X;Y) = I(X;Y). \quad (24)$$

To understand the utility of these three variables in a better way observe Table 1. This table shows the spectrum of dependence between the input and output symbol sets in terms of information variables by taking three different cases into consideration.

Table 1. Range of Values of the Key Variable in Figure 4

Level of Dependency between input X and output Y

Low Dependency ⟷ High Dependency

Variables	X and Y are independent	X and Y are Related somewhat	X and Y are highly correlated
H(X)	H(X)=same	H(X)=same	H(X)=same
H(Y)	0	0<H(Y)<H(X)	H(Y)=H(X)
H(X/Y)	High	Medium	0=Low
H(Y/X)	High	Medium	0=Low
H(X,Y)	High	Medium	Low=H(X)=H(Y)
I(X,Y)	0	0<I<H(X)	I=H(X)=H(Y)
D_R(X,Y)	High	Medium	0
E_f(X,Y)	0	$0<E_f<1$	1
	Case 1	Case 2	Case 3

Case 1: The received symbols $\underline{Y}$ are independent of the input symbols $\underline{X}$.
Case 2: The received symbols $\underline{Y}$ are somewhat related to input symbols $\underline{X}$ but both the conditional entropies are non-zero.
Case 3: The received symbols are precisely equal to the input symbols $\underline{X}$.

We will look into these cases a little more carefully in the following section.

4.4.1 Discussion of Entries in the Table 1

H(X) is the input to the information channel, which is assumed to be constant; *H(X|Y)* is the loss in bits/sec from *H(X)* to the environment which can never be recovered; *I(X;Y)* is the mutual information; *H(Y|X)* is the spurious uncertainty; and *H(Y)* is the received level of uncertainty at the channels output.

The ranges of D_R and E_f are very interesting. When the random variables X and Y are completely independent, mutual information between them is zero. So D_R is at its maximum and at the same time E_f will be zero. When X and Y are 100% correlated then D_R will be zero and E_f will be equal to 1, its largest value. So, the information channel is maximally efficient in producing the information flow. When X and Y fall between the extremes of being totally independent or totally correlated, then D_R is a positive number ($0 < D_R < D_{Rmax}$) indicating the relative distance between the random variables, whereas E_f has a value between 0 and 1 which reflects the percentage of information flowing in relation to its original input *H(x)* and it is normalized, accordingly.

5.0 APPLYING GRAPH AND INFORMATION THEORY CONCEPTS TO COMPLEX NETWORKS

Graph theory is an important tool for the analysis of the structure of certain network centric systems. Optimization problems have to be formulated to deal with actual flows and bottleneck situations. The maximum or minimum flow problem [2, 3] may be formulated in graph theory terms as follows:

$$\text{Maximize the flow} = \sum_{i} (C_{ib} - C_{bi}) \tag{25}$$

$$\text{or Minimize the quantity: (cost * flow)} = \sum_{i,j} (C_{ic} C_{ib} - C_{jc} C_{jb}) \tag{26}$$

$$\text{Subject to a number of constraints such as: } C_{ia} \geq C_{ib},\ C_{ib} \geq 0, \text{ etc.} \tag{27}$$

Then an algorithm could be obtained to calculate the flow.

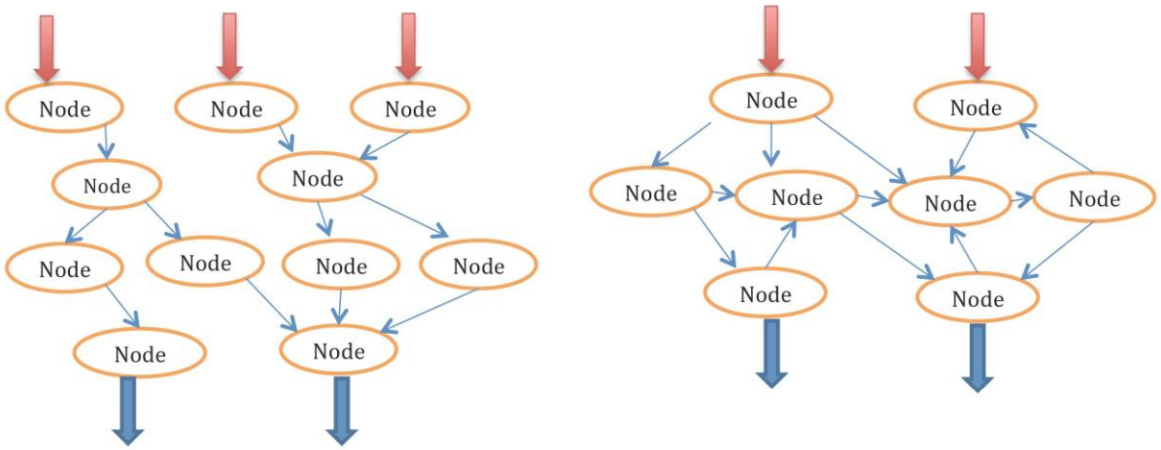


Figure 5. Two Network Centric Systems

Consider the above networks in Figure 5, the network on the right hand side looks simple when compared to the one on the left side. This is because the right hand side one has fewer layers and nodes compared to left hand side one. However, if we look at the networks closely we can observe that the network on the right hand side has three feedback links, which decreases the efficiency and the flow capability of the network. Thus, the inefficiency of flows is a structured attribute and may not easily be discerned from the adjacent and incident matrices of the graph theory. To determine flow in an objective and quantative manner, other means of discussion are required such as one that occurs in information theory.

The end goal of this discussion is that we should be able to identify the part of the network where congestion occurs. This would be very helpful for decision makers such as commanders. After these congestion areas can be identified in a spatial sense, then the changes can be attempted, such as adding or subtracting nodes/links. The above discussion can be illustrated in Figure 6.

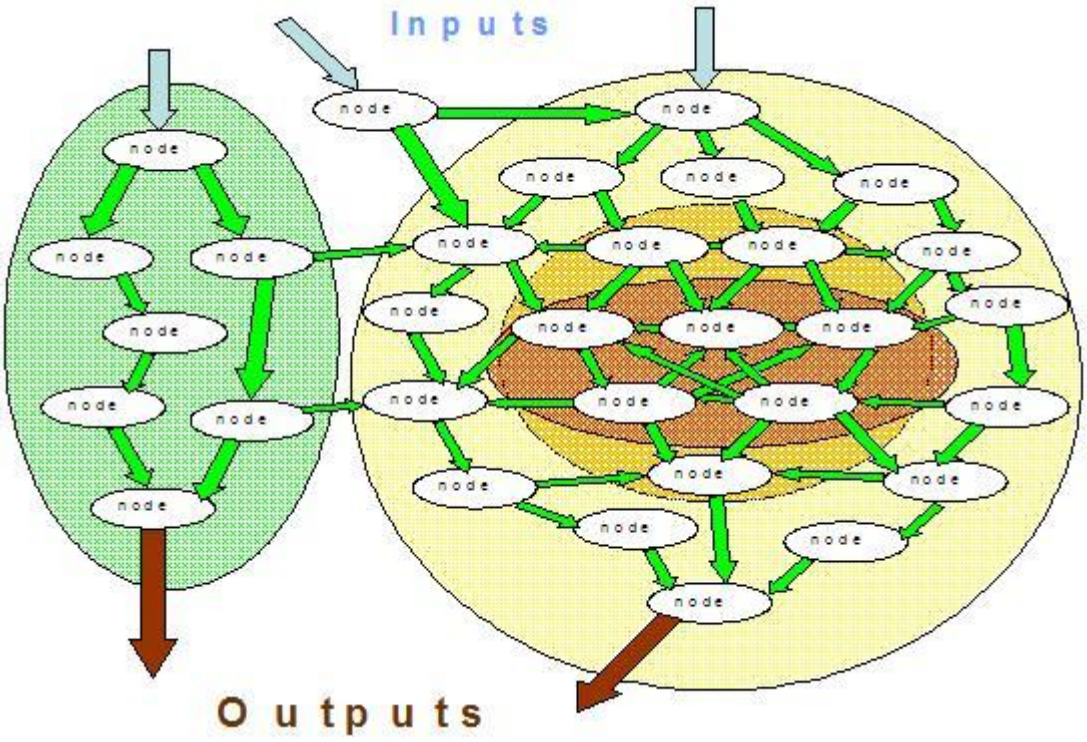


Figure 6. Visual Rendering of the Different Flow Characteristics of Distributed Systems

In Figure 6, it can be seen that different regions of the graph experience different local flow and congestion issues. Developing mathematical methods to identify these issues is an important goal.

The approach taken in this report combines the techniques from three disciplines: graph theory, optimization theory and information theory. Figure 7 explains how these disciplines interact. What is required next is to convert a complex network into an information theoretic framework. The following section explains this in detail.

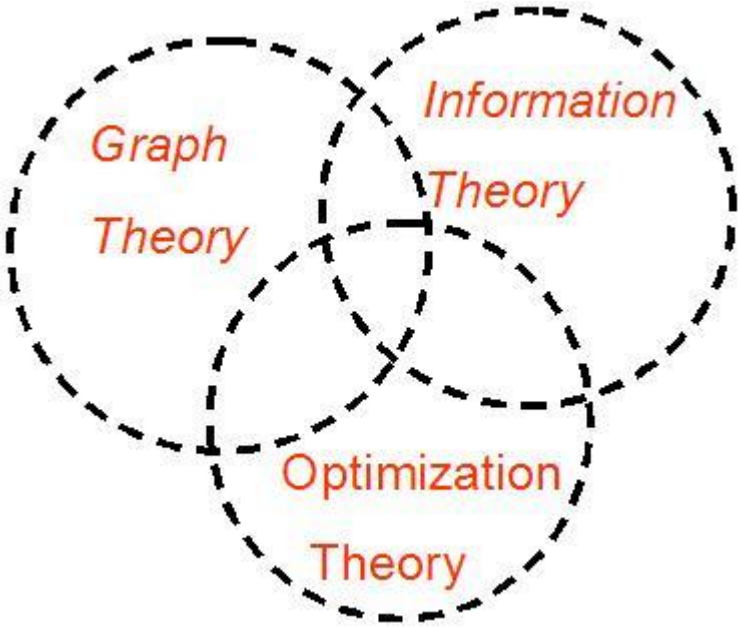


Figure 7. Combining Three Major Disciplines

5.1 Applying Information Theoretic Methods to the Network Centric System:

The goal here is to quantify the following properties of complex networks:

(1) Their congestion characteristics being measured via a flow variable.
(2) Develop an objective measure of a network's vulnerability.

A complexity measure for a distributed networks developed by Repperger [12] is briefly presented here. A complexity impedance measure is defined as follow:

$$\text{Complexity measure} = [\alpha/(\det(\overline{C}_n^T \overline{C}_n))] \log_2(H(x)/H(y)) \tag{28}$$

where α = constant of proportionality.
$\overline{C}_n$ = contingency matrix which will be defined from information theory models for specific intricate configuration of networks, and
H(x) and H(y) will also be described within the context of network-centric systems.

To determine the flow in a network is the key. The following will show the procedure to convert a network in a paradigm in which the actual flow in an ellipsoid can be calculated using information theory techniques. With no sources or sinks inside the ellipse, it should be noted that the closed cut set requires

$$\sum flows = 0$$

or

$$I_1 + I_2 + I_3 = O_1 + O_2 . \tag{29}$$

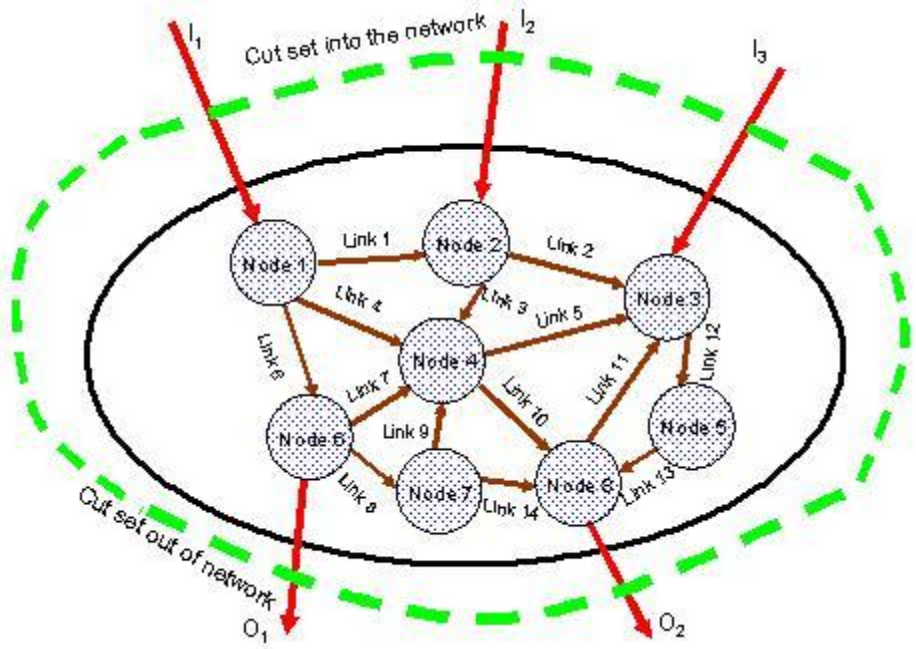


Figure 8. Framework for Combining Graph Theory with Information Theory

We make the following assumptions to implement these procedures:

$[\alpha/(\det(\overline{C}_n^T \overline{C}_n))] \log_2(H(x)/H(y))$.

Assumption 1
The above figure is an ellipsoid isolated from a more intricate network system. We treat this ellipsoid as an information channel.

The three inputs I_1, I_2, and I_3 represent the vector input flow and these can be viewed as input symbols to the information channel. Similarly O_1 and O_2 are the output flows, which can be viewed as received symbols. For any arbitrary ellipse, they are either arrows that enter the ellipse or arrows that leave the ellipse, so they can be classified as either inputs or outputs with no internal source or sinks.

Assumption 2
The net sum of the flows across the cut set would sum to zero with no source or sink inside an ellipsoid, much like Kirchhoff's current law into a node of a circuit.

Assumption 3
The nodes are connected by links, which are characterized by the triplet $[C_{ja}, C_{jb}, C_{jc}]$, where C_{ja} is the link capacity, C_{jb} is actual link flow (load) and C_{jc} is the cost. Naturally $C_{jb} \leq C_{ja}$. The total cost of a particular flow can be given by $C_{jc}C_{jb}$, which will represent an important performance parameter. The cost C_{jc} may have units of time or resources.

Assumption 4
A contingency table and contingency matrix are necessary to analyze a network in an information theoretic sense. To define C_n, a response matrix must be synthesized in communication systems and information theory [12,13]. In communications, a response matrix helps in identifying all the actions of responses necessary to complete a task, where as in complex networks it helps in translating the total

input flows required to produce a necessary output. The following examples show that the response matrix can be described as it occurs commonly in communication systems.

Assumption 5
A contingency table is formulated by normalizing all the elements of the response matrix.

Assumption 6
With the contingency table constructed from assumptions 4 and 5, the entropies *H(X)*, *H(Y),* and *H(X,Y)* can be determined as follows:

(a) *H(X)* is the sum across all the rows.
(b) *H(Y)* is the sum down all the columns.
(c) *H(X,Y)* is the sum of all the elements in the table.

The calculation of the remaining variables easily proceeds since:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{30}$$

is a true flow variable. The remaining two entropies can then be determined:

$$H(X/Y) = H(X) - I(X;Y) \tag{31}$$

$$H(Y/X) = H(Y) - I(X;Y)\ . \tag{32}$$

Six examples are now worked to show how to apply this method to arbitrary networks.

5.2 Six Examples to Illustrate the Procedure

We now consider six examples to show the procedure. Our goal is to examine the complexity impedance and other flow parameters of all the networks. In the first four examples, only forward flows are considered. However in Examples 5 and 6 backward flows are taken into account because of which we are going to observe an increase in complexity impedance. For the sake of simplicity, all flow variables will be denoted as f_i.

***Case 1*:**
Figure 9 is just a concatenation of simple forward flow through two nodes in series. The first element in the response matrix is sum of all flows entering node 1. This is f_1. The next element is given by flows from node 2 to node 1. The last element in the matrix is the sum of all forward flows into node 2 which is f_1. To make the contingency matrix, normalize the elements of response matrix. This is done by dividing every element by sum of all elements in response matrix. Summing all the flows response yields

$$f_1 + f_1 + f_1 = 3f_1.$$

Therefore contingency table contains one-third as three elements. The contingency table is shown in Figure 9. Calculations for entropy and mutual information are shown below.

On the right side of the contingency table we sum across the columns for each row. This determines the elements for *H(X)*. We then sum each column and place those sums at the bottom of the contingency table to calculate *H(Y)*. Then we calculate *H(X,Y)* for every element in the contingency table.

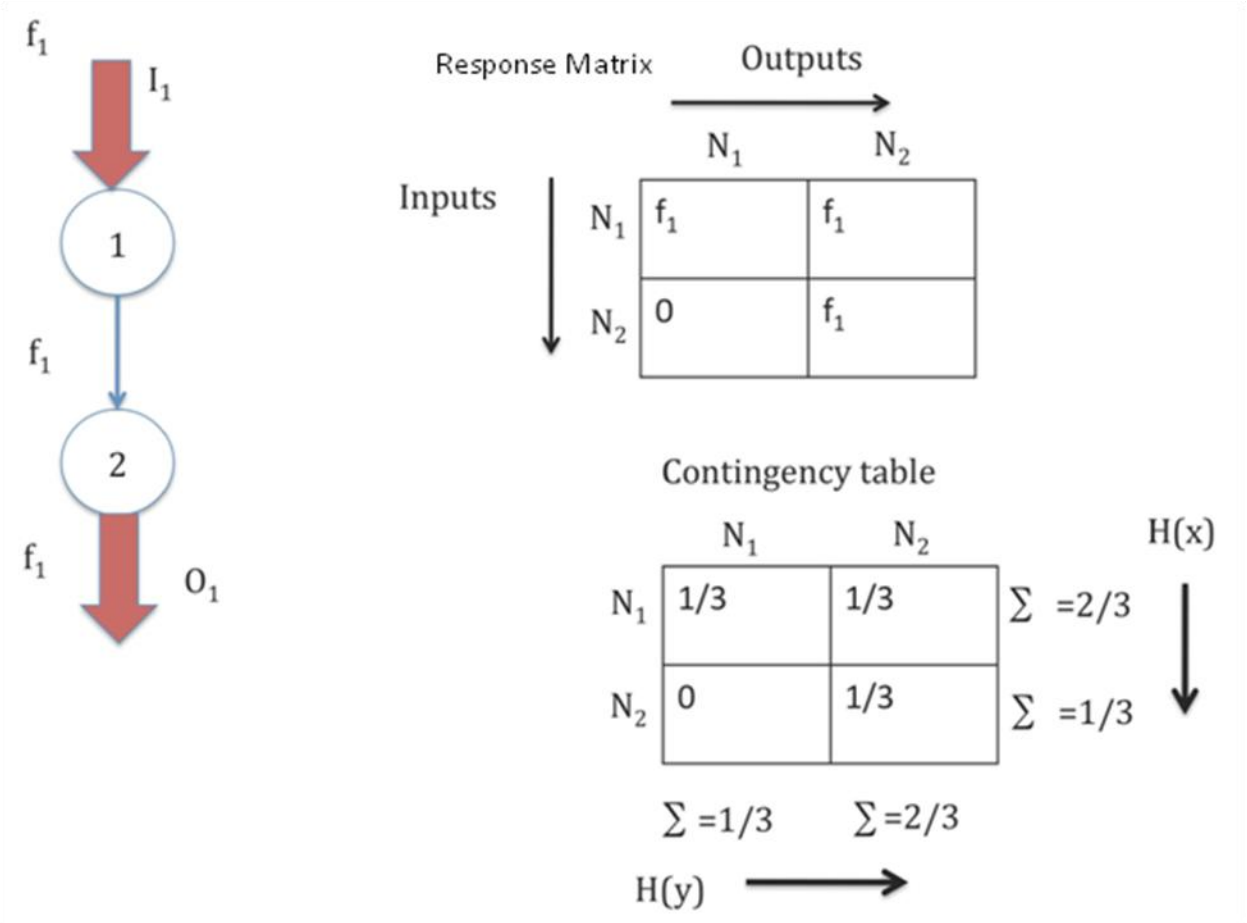


Figure 9. Simple Forward Flow describing Case 1

$$H(X) = (2/3)\ \log_2(1/(2/3)) + (1/3)\ \log_2(1/(1/3)) = 0.9183 \text{ bits} \quad (33)$$

$$H(Y) = (1/3)\ \log_2(1/(1/3)) + (2/3)\ \log_2(1/(2/3)) = 0.9183 \text{ bits} \quad (34)$$

$$H(X, Y) = (1/3)\ \log_2(1/(1/3)) + (1/3)\ \log_2(1/(1/3)) + (1/3)\ \log_2(1/(1/3)) = 1.5850 \text{ bits} \quad (35)$$

Then

$$I(X; Y) = H(X) + H(Y) - H(X, Y) = 0.9183 + 0.9183 - 1.5850 = 0.2516 \text{ bits} \quad (36)$$

$$\text{and } H(X / Y) = H(X) - I(X; Y) = 0.9183 - 0.2516 = 0.6667 \text{ bits} \quad (37)$$

$$\text{with } H(Y / X) = H(Y) - I(X; Y) = 0.9183 - 0.2516 = 0.6667 \text{ bits.} \quad (38)$$

This completes the Shannon variables. The remaining calculations are determined via:

$$D_R = H(X / Y) + H(Y / X) = 0.6667 + 0.6667 = 1.3333 \text{ bits} \quad (39)$$

$$Ef = (I(X; Y)) / H(X) = 0.2516 / 0.9183 = 0.2740 \text{ (dimensionless)} \quad (40)$$

***Case 2*:**

As discussed earlier, rows represent outputs and columns represent the inputs. The first element in the first row is given by the total number of flows into node N_1 which is equal to $f_1 = 6$. The second element in that row is given by total flow from N_1 to N_2 which is equal to $f_{a1} = 3$. Similarly, the third element is given by $f_{a2} = 3$. The first element in the second and third row is equal to zero because there is nothing

going from node 2 or node 3 to node 1. Similarly, the second element in third row is equal to zero. The last element in the third row is given by sum of the flows getting into node 3 which is given by $f_b + f_{a2} = 6$. If we closely observe the figure we can see that the network discussed below satisfies the second assumption discussed earlier. We can observe this in Figure 10.

$$H(X) = -(12/24)\log_2(12/24) - (6/24)\log_2(6/24) - (6/24)\log_2(6/24) = 1.50 \text{ bits} \quad (41)$$

$$H(Y) = -(6/24)\log_2(6/24) - (6/24)\log_2(6/24) - (12/24)\log_2(12/24) = 1.50 \text{ bits} \quad (42)$$

$$\text{and} \quad H(X, Y) = -2(6/24)\log_2(6/24) - 4(3/24)\log_2(3/24) = 2.50 \text{ bits.} \quad (43)$$

$$\text{Again:} \quad I(X; Y) = H(X) + H(Y) - H(X, Y) = 1.50 + 1.50 - 2.50 = 0.50 \text{ bits} \quad (44)$$

$$\text{and } H(X / Y) = H(X) - I(X; Y) = 1.50 - 0.50 = 1.0 \text{ bits} \quad (45)$$

$$\text{with } H(Y / X) = H(Y) - I(X; Y) = 1.50 - 0.50 = 1.0 \text{ bits .} \quad (46)$$

Thus the remaining calculations are determined via:

$$D_R = H(X / Y) + H(Y / X) = 1.0 + 1.0 = 2.0 \text{ bits} \quad (47)$$

$$E_f = (I(X; Y)) / H(x) = 0.5 / 1.5 = 0.3333 \text{ (dimensionless) .} \quad (48)$$

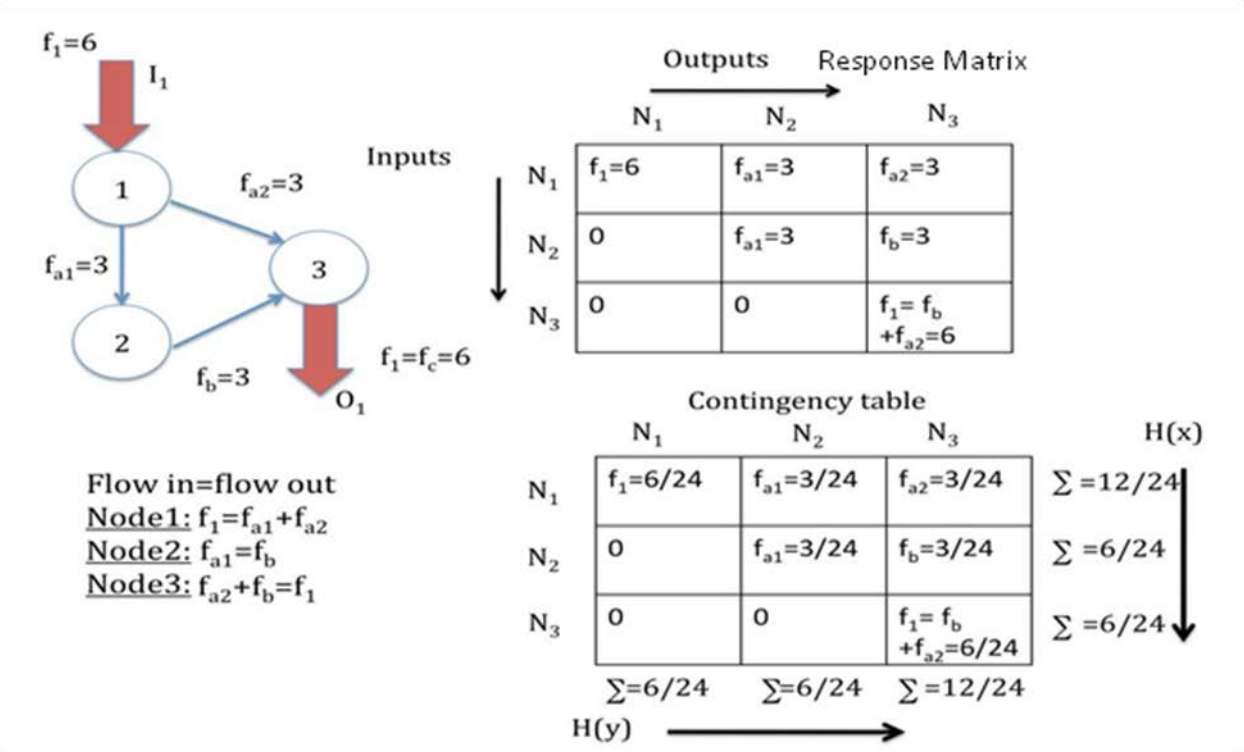


Figure 10. Forward Flow describing Case 2

***Case 3*: (Complex forward flow with no impedance)**

If we observe the response Table for Case 2, we can say that the total number of input flows into a node gives all the diagonal elements. Off diagonal element R_{ij} is given by the flow i^{th} node to the j^{th} node in the network. The network in Figure 11 looks a bit complex but does not have any impedance because it did not have any feedback loops.

$$H(X) = 1.9159 \text{ bits}; \; H(Y) = 1.9159 \text{ bits} \tag{49}$$

$$H(X,Y) = -2(9/45)\log_2(9/45) - 2\,(3/45)\log_2(3/45) - 3(2/45)\log_2(2/45) - (5/45)\log_2(5/45) - (8/45)\log_2(4/45) = 2.687 \text{ bits} \tag{50}$$

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = 1.9159 + 1.9159 - 2.687 = 1.1448 \text{ bits} \tag{51}$$

$$H(X/Y) = H(X) - I(X;Y) = 1.9159 - 1.1448 = 0.7711 \text{ bits} \tag{52}$$

$$H(Y/X) = H(Y) - I(X;Y) = 1.9159 - 1.1448 = 0.7711 \text{ bits}. \tag{53}$$

The remaining calculations are calculated using

$$D_R = H(X/Y) + H(Y/X) = 0.7711 + 0.7711 = 1.5422 \text{ bits} \tag{54}$$

$$E_f = (I(X;Y)) / H(X) = 1.1448 / 1.9159 = 0.5975 \text{ (dimensionless)}. \tag{55}$$

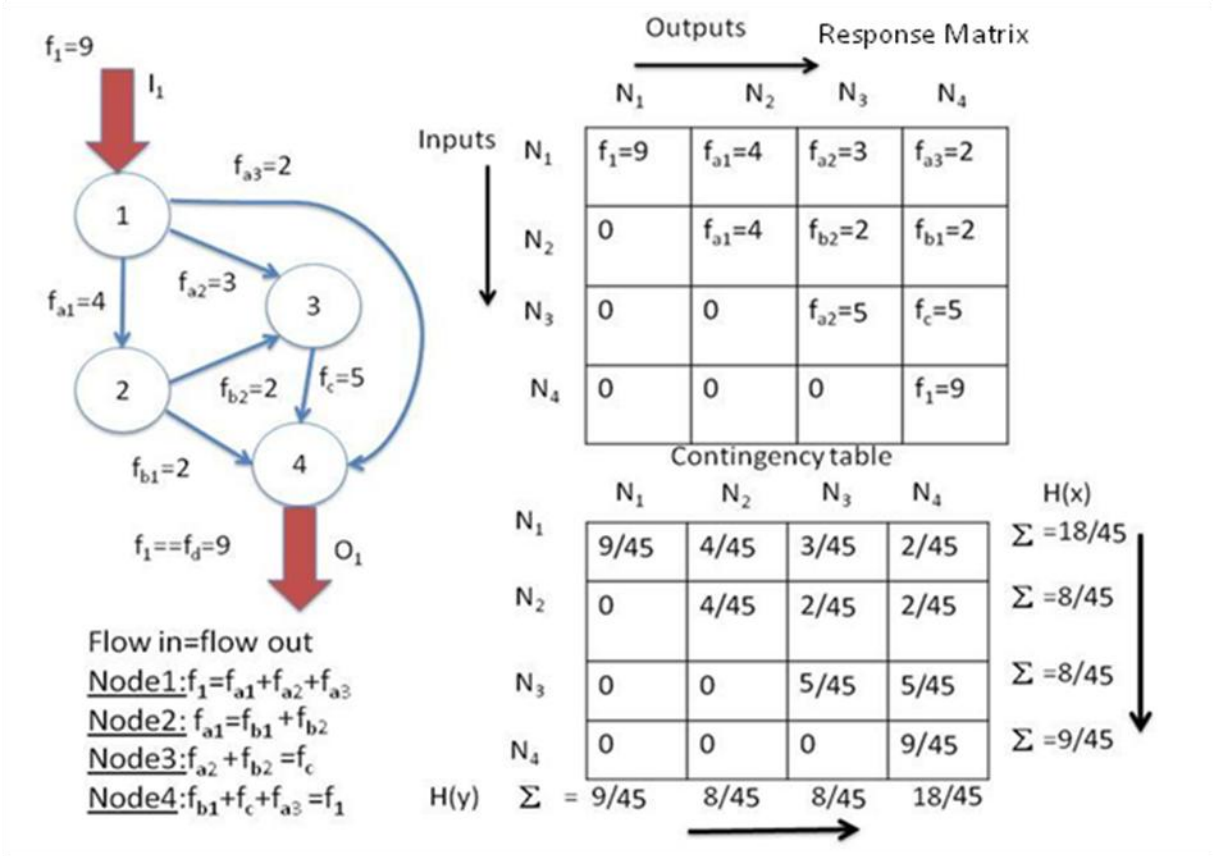


Figure 11. Complex Forward Flow with No Impedance as described in Case 3

***Case 4*: (Single feedback with impedance)**

If we observe the response tables from the above example, we see that the off diagonal elements present on the left side of the matrix are all zeroes. This is because there is no feedback loops. Now we consider a feedback loop f_{b1} from node 2 to node 1. So, now we have the first element in the second row which is equal to $f_{b1}= 1$ (observe that the rest of the elements are all zeroes because of the absence of feedback loops). The rest of the table is calculated just like the above cases. The response matrix and Contingency matrix are shown in Figure 12.

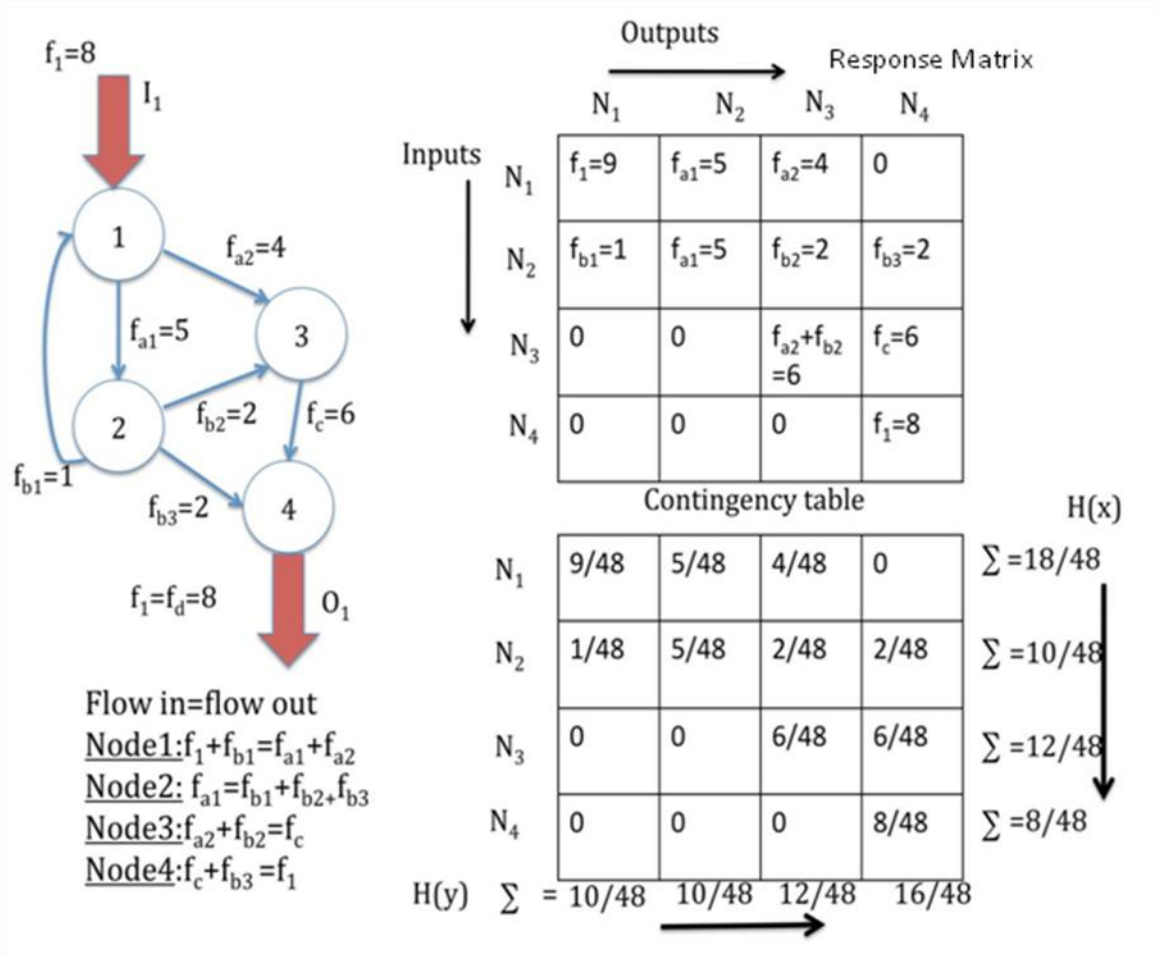


Figure 12. Single Feedback with Small Impedance as described in Case 4

***Case 5*: (Double feedback with impedance)**

Another feedback loop f_{d1} is added to the above example. The input loop flowing to node 2 are kept constant as compared to above two cases in for consistency. Also, only the positive directed arrows indicating flows are calculated in the diagonal elements of the response matrix. Again, the off diagonal elements represent the flows out of the respective nodes. The network and the contingency table for this case is shown in Figure 13.

Case 6: (Fully Connected and Maximum Impedance):

In Figure 14, a fully connected graph is displayed. Again, the same output flows of Node 2 are kept constant with Cases 3-5, for consistency. This digraph represents one of the most complex forms of feedback. This graph has a flow arrow from each node to every other possible node. There are a total of six feedback loops: f_{b1}, f_{c1}, f_{c2}, f_{d1}, f_{d2}, and f_{d3}.

The results of these six cases are summarized in Table 2 (α=1 in equation (27)) by plotting the key variables D_R, E_F and I, complexity impedance.

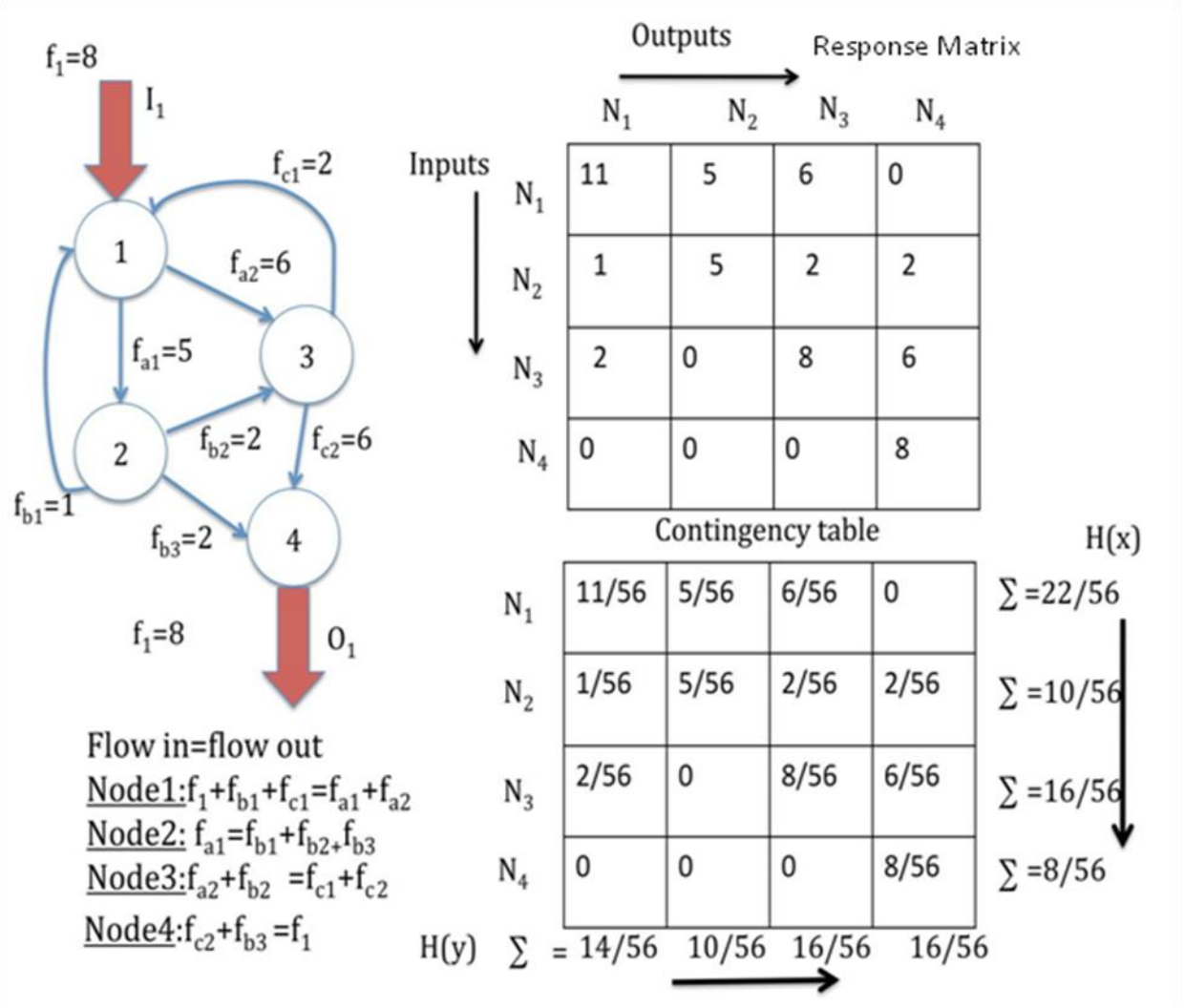
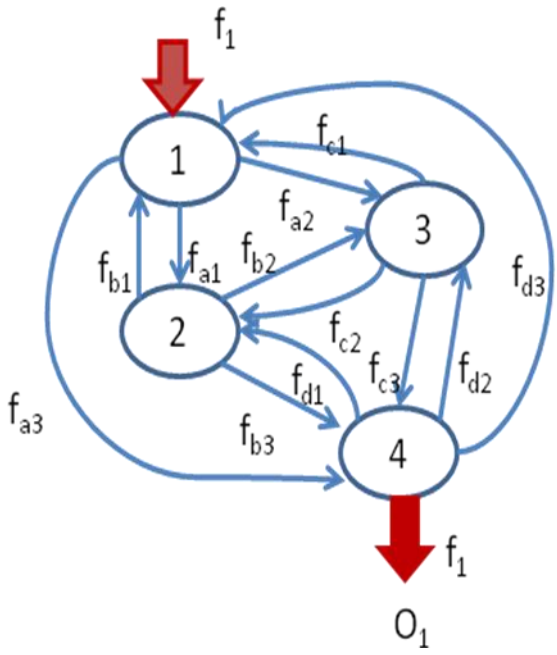


Figure 13. Double Feedback with Impedance as described in Case 5



Response Matrix Outputs

Inputs

	N_1	N_2	N_3	N_4
N_1	$f_1+f_{b1}+f_{c1}+f_{d3}=8$	$f_{a1}=3$	$f_{a2}=4$	$f_{a3}=1$
N_2	$f_{b1}=1$	$f_{a1}+f_{c2}+f_{d1}=5$	$f_{b2}=2$	$f_{b3}=2$
N_3	$f_{c1}=1$	$f_{c2}=1$	$f_{a2}+f_{b2}+f_{d2}=7$	$f_{c3}=5$
N_4	$f_{d3}=1$	$f_{d1}=1$	$f_{d2}=1$	$f_{c3}+f_{b3}+f_{a3}=8$

Sum of responses = 51

$f_{a1}=3$, $f_{a2}=4$, $f_{a3}=1$,
$f_{b1}=1$, $f_{b2}=2$, $f_{b3}=2$,
$f_{c1}=1$, $f_{c2}=1$, $f_{c3}=5$,
$f_{d1}=1$, $f_{d2}=1$, $f_{d3}=1$,
$f_{d4} = f_1 = 5$

Flow in = Flow out

Node 1: $f_1+f_{b1}+f_{c1}+f_{d3}=f_{a1}+f_{a2}+f_{a3}$

Node 2: $f_{a1}+f_{c2}+f_{d1}=f_{b1}+f_{b2}+f_{b3}$

Node 3: $f_{b2}+ f_{a2}+ f_{d2}= f_{c1}+ f_{c2}+ f_{c3}$

Node 4: $f_{a3}+ f_{b3}+ f_{c3}= f_{d1}+ f_{d2}+ f_{d3}+ f_1$

Contingency Table

	N_1	N_2	N_3	N_4	H(x) $\sum=$
N_1	8/51	3/51	4/51	1/51	16/51
N_2	1/51	5/51	2/51	2/51	10/51
N_3	1/51	1/51	7/51	5/51	14/51
N_4	1/51	1/51	1/51	8/51	11/51
$\sum=$	11/51	10/51	14/51	16/51	

H(y)

Figure 14. Fully Connected and Maximum Impedance as described in Case 6

Table 2. Calculations of Cases 1-6 in Figures 9-14

Case	*H(X) Bits*	*H(Y) Bits*	*H(X,Y) Bits*	*I(X;Y) Bits*	*H(X/Y) Bits*	*H(Y/X) Bits*	D_R *Bits*	E_f	$\det(\overline{C}^T * \overline{C})$	Complexity Impedance
1	.918	0.92	1.59	0.25	0.667	0.667	1.33	.27	1.23×10^{-2}	0
2	1.50	1.50	2.50	0.50	1.000	1.000	2.0	0.33	6.1×10^{-3}	0
3	1.915	1.915	2.687	1.144	0.771	0.771	1.542	0.597	1.56×10^{-7}	0
4	1.932	1.970	3.109	.794	1.138	1.176	2.314	.410	8.575×10^{-7}	1.877×10^{5}
5	1.890	1.976	3.193	.673	1.216	1.303	2.519	.356	1.308×10^{-7}	4.894×10^{5}
6	1.97	1.934	3.619	.2889	1.685	1.645	3.331	.1463	5.045×10^{-8}	5.985×10^{5}

What is interesting in Table 2 is the monotonic increase of information variables with the number of feedback loops. The complexity impedance and D_R are increasing with the number of feedback loops whereas I (X; Y) and E_f are decreasing.

Interpretation of I (x; y):

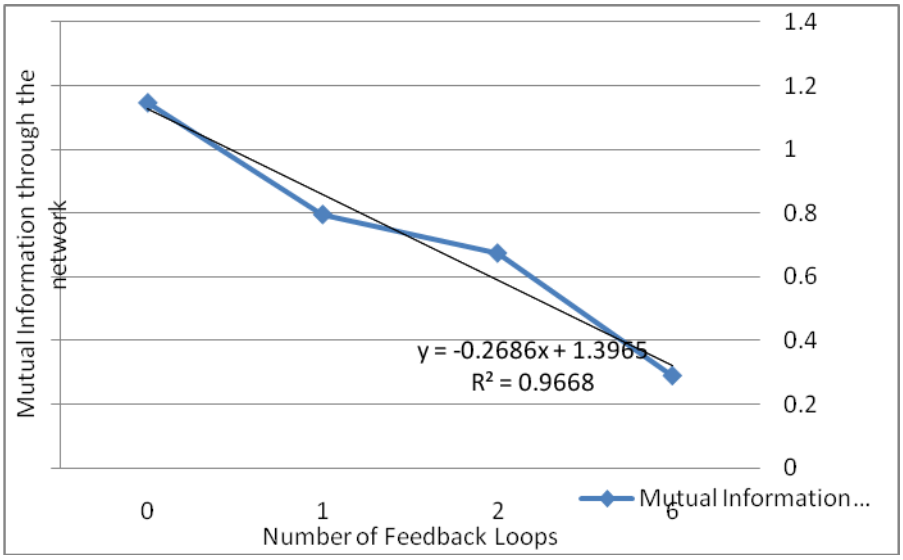


Figure 15. Mutual Information (and Linear Regression) as the Number of Feedback loops Increases

Our goal is to render specific bottleneck areas in the network. If we draw an ellipsoid over certain part of a network, we can find I(X; Y) in the region. Since I(X; Y) has units of bits, the average throughput flow in a given ellipsoid is given by $2^{I(X;Y)}$. It is very clear that I(X; Y) has been affected by the addition of feedback loops. A linear regression through the data collected shows with 97% regression correlation that the calculated I(X; Y) is almost linearly decreasing with number of feedback loops.

6.0 APPLICATION TO A LOGISTIC DISTRIBUTED NETWORK SYSTEM

Logistics is an area with a rich history of distributed network investigations. A brief overview of logistics is presented to show the relevance to the complex network problems of interest in this work.

6.1 Brief Overview of Logistic Related Applications

In the military, logistic applications are used in Battle Field Ad Hoc Networks to study reliable access to services and resources in future combat systems. For combat logistics, a flow model has been developed to plan battles with minimum inventory. Logistics is used in mobile mesh networks to provide survivable and reliable means of gaining information superiority and support efficient minion execution [14]. Team munitions logistics are simulated in a Brigade Combat Team study via a discrete event simulation model to examine a number of different operating conditions and to understand the sensitive parameters in a better way for a successful operation [15]. In [16,17] a logistics system which emulates a Computer based Aerial Port Simulation, CAPS (cf., Figure 16) is investigated for the problem of refreshing or resupplying an aircraft just landing on the ground. Many issues related to human collaboration and team interactions can be studied within this platform. This specific application gives us an opportunity to examine network centric systems, especially when they are distributed in nature.

Formulating a flow problem and solving the requisite optimization problem by a procedure such as a genetic algorithm can address the optimization of flows in a logistics system [18-21]. There are several other optimization techniques that are used in logistics such as the simplex method [22-25]; even graph procedures can be used for this purpose. Other means of providing performance improvement in a logistics network encompass efficiency measures including variables such as time performance and other productivity measures. The best way to address the improvement of a logistics system is enhancing the supply chain parameter [26-31]. The study of information and information systems is a natural extension of these concepts [32]. The information capacity of a network is a byproduct of the idea. If we look into the literature of logistics we can find a variety of eclectic topics such as flow shop problems, which are computationally difficult [33].

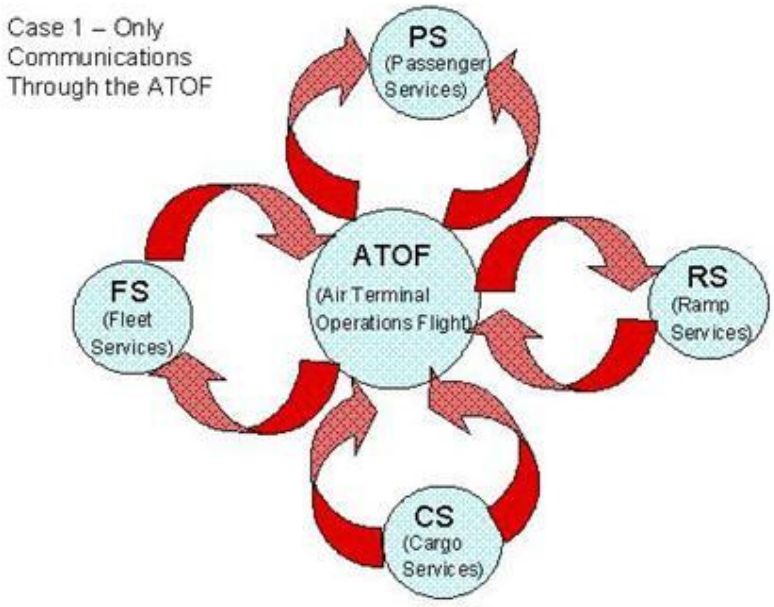


Figure 16. The CAPS (Computer-based Aerial Port Simulation) Logistics Network

Another important observation is that we see a great deal of exploitation of large-scale networks and their dynamic behavior. A diverse set of applications was noted such as in the lumber industry looking at the trust between humans as they distort information flow along the supply chain [34] and survivability of such complex systems [35].

A brief survey of the logistic literature indicates the need and interest in enhancing the understanding of performance of such distributed network systems. The focus of this report is in the US Air Force Military Services. As previously mentioned, this work will expand upon the simulation in [16,17] to show how to apply the three disciplines of information, graph and optimization theories to better understand flow performance in logistics systems when viewed within the context of a distributed network-centric system.

6.2 Modeling the Network of the Logistics Distributed Network:

A logistics Distributed Network System is modeled as in Figure 16 to show how to implement the method of interest. In Figure 16, we consider a particular situation where communication is only done through an Air Terminal Operations Flight, ATOF. In Figure 17, an alternative condition is considered in which all possible redundant communications are allowed across all constituent players. This second condition represents a ―full spam" situation, which includes an unnecessary amount of communications. In Figure 17, the thick arrows represent redundant communications, which are outside the scope of the performance of the minion. The roles of the key players are defined as follows:

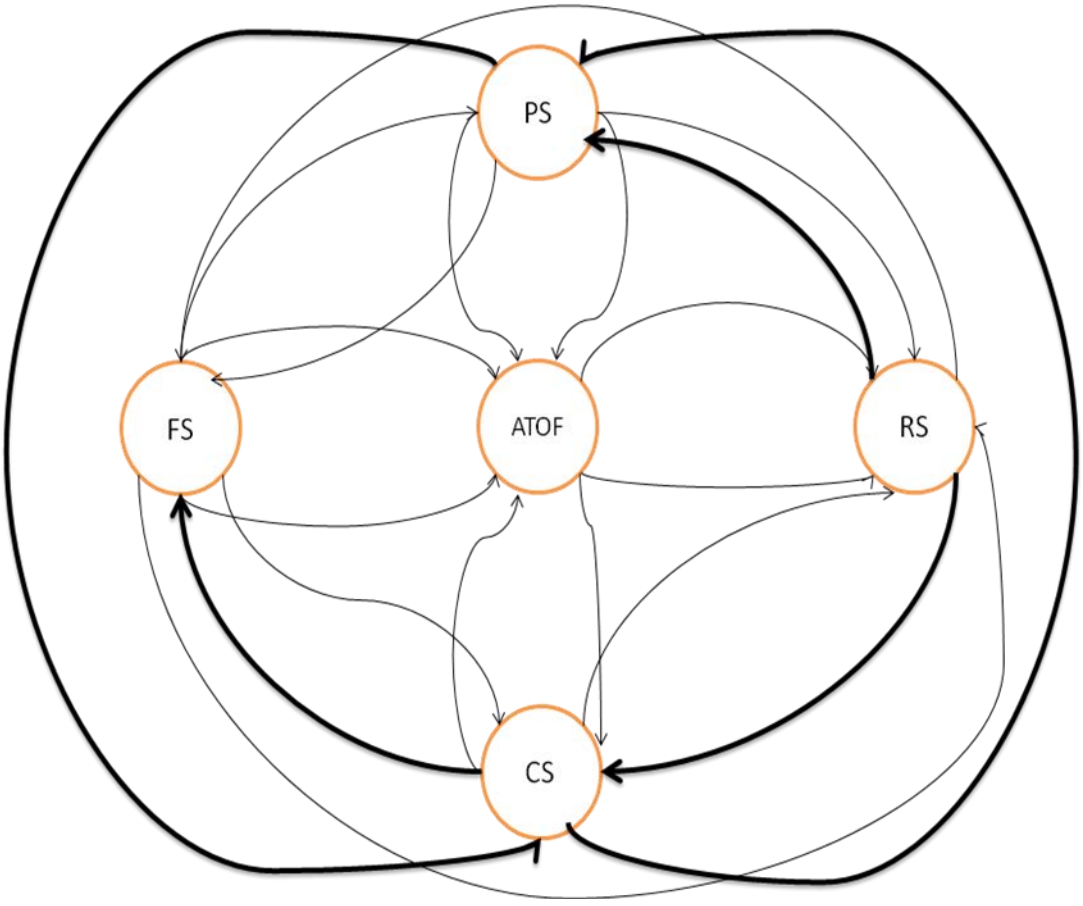


Figure 17. Full Spam Communications

ATOF (Air Terminal Operation Flight):
The key role of this individual is to direct and monitor the remaining four members in the network. Initially, as an aircraft lands and is prepared for takeoff, it is taken to a specific ramp location. The numbers of seats available, the destination and cargo characteristics have to be determined and communicated to the key players. Providing proper temporal sequence order of operations of constituent players is as important as monitoring their operation. One important safety constraint is that there should be no passenger in the plane when loading or unloading of cargo is taking place.

PS (Passenger Services):
After the ATOF gives the information about the ramp location and destination, the passenger‘s list is checked and manifest is determined for the departing aircraft list. The old passengers have to be deplaned and the new passengers have to be boarded when directed by ATOF.

CS (Cargo Services):
The cargo must be determined specific to the next destination location. This cargo is sequenced and put in explicit bins to facilitate ramp services.

RS (Ramp Services):
This network member must unload the inbound cargo and outbound cargo subject to constraints such as no passengers onboard during these activities performed in the proper temporal sequence.

FS (Fleet Services):
The duties include cleaning the aircraft after the passengers and inbound cargos are removed. They also include the uploading of supplies and the correct number of meals required after the outbound passengers have been put on the plane before takeoff.

In the case as shown in Figure 16, communications are directed only through ATOF, which is simple, but there is no spam. All communications are pertinent. In Figure 17, every network member communicates with the rest of the members. The spam condition is in full effect. This represents more work, more unnecessary disruptions with communications. However, the reliability and reduction of vulnerability of the network may be improved by these network conditions.

7.0 FORMULATION OF AN OPTIMIZATION PROBLEM TO AMELIORATE THE FLOW

Figure 18 is a modified rendering of Figure 16 as a graph representation that generalizes the case in which communications are done only through ATOF with minimum but pertinent communications required. The topic of interest here is the determination of optimal flow through the overall system. This is calculated using information theory techniques.

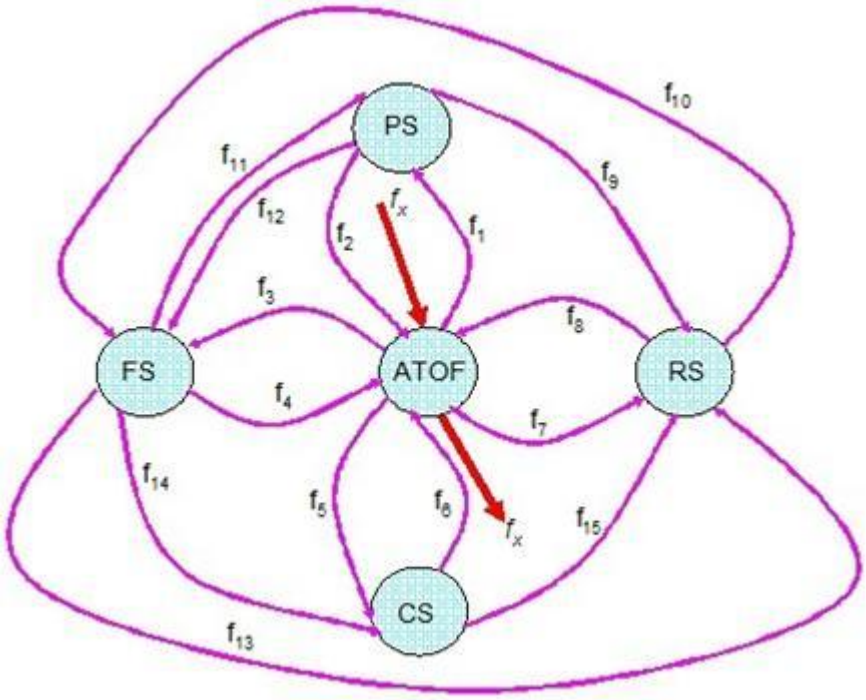


Figure 18. Graph Theory Representation of the CAPS System

In analyzing the graph in Figure 18, the presumption is that ATOF is influenced by a source f_x and since the other players (PS, RS, FS, CS) do not interact with either a source or sink, they obey Kirchhoff's law (the second assumption in Section 5) in cut sets for flow into and out of each vertex or node. The term "f_x" here represents the number of aircrafts that need to be served at a given time. Also, flow cost along a line or arc will be denoted by C_{jc}. Thus the optimization problem can be formulated as follows:

Select f_i where i=1 to 15 in Figure 18 such that either (a) or (b) is true.

(a) A maximum of $I(x;y)$ is obtained for a fixed f_x input.

(b) Or a minimum of J_1 is obtained, where $$J_1 = \sum_{i=1}^{15} f_i \, C_{ic} \tag{56}$$

subject to the following flow constraints(as a result of Kirchoff's Cut Set Laws at each node):

ATOF: $f_x + f_2 + f_4 + f_6 + f_8 = f_1 + f_3 + f_5 + f_7 + f_x$ (57)

PS: $f_{11} + f_1 = f_9 + f_2 + f_{12}$ (58)

RS: $f_{15} + f_7 + f_9 + f_{13} = f_{10} + f_8$ (59)

FS: $f_{10} + f_{12} + f_3 = f_{14} + f_{13} + f_4 + f_{11}$ (60)

CS: $f_5 + f_{14} = f_{15} + f_6$ (61)

These are 15 unknowns with five constraints leaving 10 dependent variables. We can express the five equations in matrix form as shown below

$$A_{5x15}\, X_{15x1} = 0_{5x1} \tag{62}$$

$$\begin{bmatrix} -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} f1 \\ f2 \\ f3 \\ f4 \\ f5 \\ f6 \\ f7 \\ f8 \\ f9 \\ f10 \\ f11 \\ f12 \\ f13 \\ f14 \\ f15 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

If we observe the rank of matrix A, we can easily prove that only 4 out of remaining 5 are independent leaving 11 (in total) dependent variables. Thus from the 15 unknown flows 11 dependent flows need to be determined. Some other approaches to calculate the flow are mentioned below.

7.1 Some Traditional Methods to Approach Flow Problems

The Kelly method is widely regarded as the customary method to compute the optimal flow of such complex systems using a convex optimization. A game theory method can also be applied and results are generalized to internet and other distributed protocols. Here the optimization problem is succinctly outlined.

Formulation of the Flow Optimization Problem via Convex Optimization:

The goal here is to find flows f_i to minimize the function

$$Min\ J_1 = \sum_{i=1}^{15} f_i\, C_{ic} \tag{63}$$

where C_{ic} = cost of the i-th link

subject to the Kirchhoff's law at the vertices (57-61) and possibly capacity limitations $f_i \le C_{ia}$, where C_{ia} = maximum capacity of the link.

The traditional solution of the optimization problem is to define the Lagrangian as

$$L(f,p) = J_1 - \sum_{m=1}^{5} p_m \left(\sum_{i=1}^{15} f_i - c_{ia} \right) \tag{64}$$

where traditional numerical optimization algorithms determine five Lagrangian multipliers P_m. The approach will not violate the constraints. The optimization methodology used in this report is genetic algorithms but will focus on an information theoretic viewpoint for the cost function.

7.2 Flow Optimization via Genetic Algorithm for the Distributed Network

Genetic algorithms is a powerful tool to evaluate human – machine systems, [36,37]. Genetic algorithms will be used here to determine optimal flow vectors. Some numerical problems may be extremely difficult to compute due to the combinational nature of the solution. GA is used to analyze the network congestion problem in calculating the flows. The advantages of this procedure have applicability for problems of extremely difficult numerical complexity that are considered ―NP hard" (non-deterministic polynomial meaning the total computation time is of exponential order and not of a polynomial type). The genetic algorithm method provides a viable solution in reasonable time. We consider two cases in this report.

Case 1: flow variables are positive integers
Case 2: flow variables are positive real numbers.

The GA problem for Case 1 can be formulated as below:

Step 1: From our earlier discussion, we know that there are 4 dependent and 11 independent variables. Pick 11 unknown flows f_i (i = 1, 2, 3… 11) and designate each of them as a computer word with a chromosome representation shown in Figure 19. If the assumption is then made that the input to the network f_x is bounded, e.g., $0 < f_x < 7$, then the initial choice for the flows may be a 3 bit word as displayed in Figure 19. For each of the 11 unknown flow variables, there exist up to 7 non-zero choices. Thus there will be 7^{11} different combinations of flows. If we take chromosome size as 10 bits, then we will have $(2^{10} - 1)^{11}$ possibilities which is beyond the capability of any practical computer. Thus the motivation exists to apply genetic algorithms.

***Step 2*:** The criterion for selecting the chromosomes is the fitness calculated using the I(x; y) function shown in equation (11).This is done using Matlab and the corresponding code is shown in Appendix.

***Step 3*:** A pool of 20 chromosomes of the optimal fitness values is initiated, termed the ―elite pool." The way this is done is shown in the ‗Gene_Algo.m' function and we can find this in Appendix.

***Step 4*:** New individuals are then created from the elite pool, by using the standard genetic algorithm (sGA) operations of crossover (to preserve good qualities) and mutation (to include diversity) with the candidate chromosomes. The new individuals must satisfy equations (57)-(61). This is done using Crossover and Mutation functions present in the Appendix.

***Step 5*:** The new individuals are then evaluated for their fitness and compared to the present members in the elite pool. If a new member has a better fitness value (higher or lower, depending on maximization or minimization) as compared to the least optimal member of the elite pool, the new member is added to the elite pool and the worst individual of the elite pool is removed. This is achieved using ‗sort_ga.m' function.

***Step 6*:** This process is continued until either the elite pool converges to the same flow values or if a certain number of iterations are surpassed (e.g., 10,000 iterations). The most fit (members with the optimal J1 values) are considered the final members in the pool and represent the best solution.

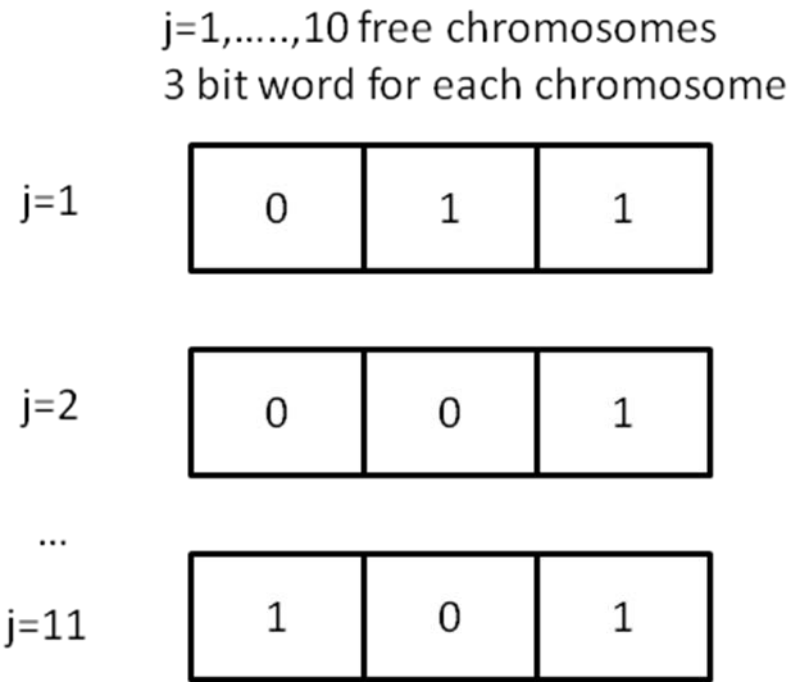


Figure 19. Generating Chromosomes for the Fitness Pool

The GA calculates optimal flows for Case 2 in the following manner.

We generate a candidate solution by doing a local search at the current best flows using a grid search technique. Generation of a candidate solution is explained below.

Generation of candidate solution:

We find the null space ‗N' of a matrix A in (62). We do our search in the null space of A. Since we have only 11 independent variables, we consider a 11- dimensional plane. The distance that we move in each of these 11 directions will be determined by vector ‗P'. Generation of ‗P' will be discussed in the next section. Once we have ‗P' we can generate the candidate solution; the code for this is presented in Appendix.

Generation of Vector 'P':

We use GA techniques to generate ‗P'. Even in this case elite pool is first initiated with same chromosomes. We apply crossover and mutation techniques to the selected chromosomes. We then generate a candidate solution from the current best one. This candidate solution is checked to examine whether these are all positive and also if it satisfies the constraints given in equation (57-61). We can find the code in the Appendix. We calculate its fitness value if it is more than the least value in the elite pool. We replace the last one with this and the fitness value is also changed. We then sort the elite pool. We continue this process until we obtain a satisfactory result or until we reach a certain number of iterations.

8.0 RESULTS AND DISCUSSIONS

In Figures 20-23 are displayed the mutual information of the best chromosome in the elite pool for maximization as well as minimization. Two cases were taken into consideration: first is the case where the flow variables are positive integers and the second case takes positive real numbers into account. Figures 20 and 21 take the first case into consideration whereas Figures 22 and 23 follow the second case. If we observe Figures 20 and 21 we can find a 390% difference (when we compare the best and worst measurements of information flows) between maximum information flow and minimum information flow. The time taken for the second case is less than that of the first; this can be due to the fact that domain for the flow variables in the second case is much larger than the first one. As we can see just by varying the values of flow variables we can achieve a huge difference in the information flowing in the network. Another important conclusion from the results would be the flow vector that maximizes information flow also maximizes the throughput performance. If we want to congest a network the best way is to manipulate the flow vector in such a way that it would minimize the information flow in the network. If the goal is to disable a network, this produces the maximum congestion as a way to degrade the information flow.

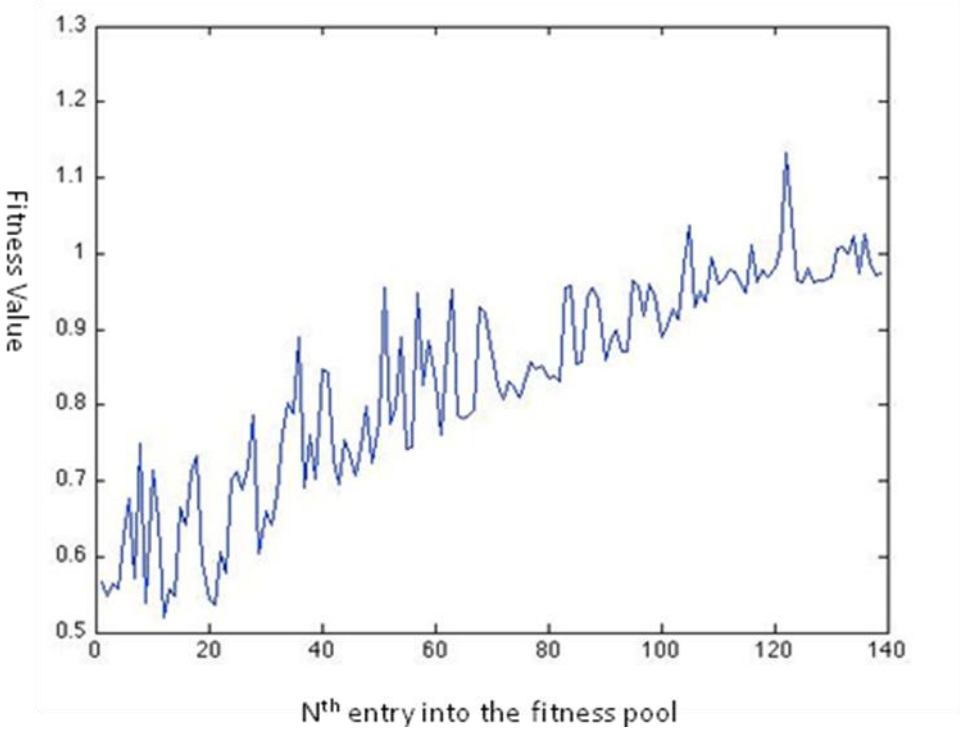


Figure 20. Maximization of I(X:Y) (flow variables – positive integers)

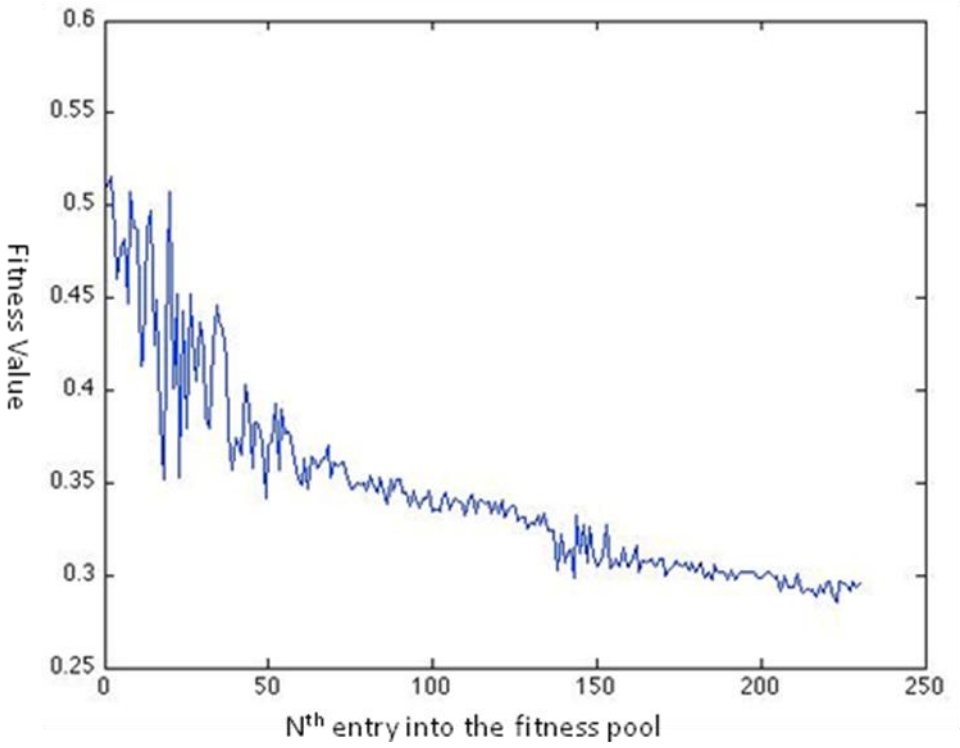


Figure 21. Minimization of I(X:Y) (flow variables – positive integers)

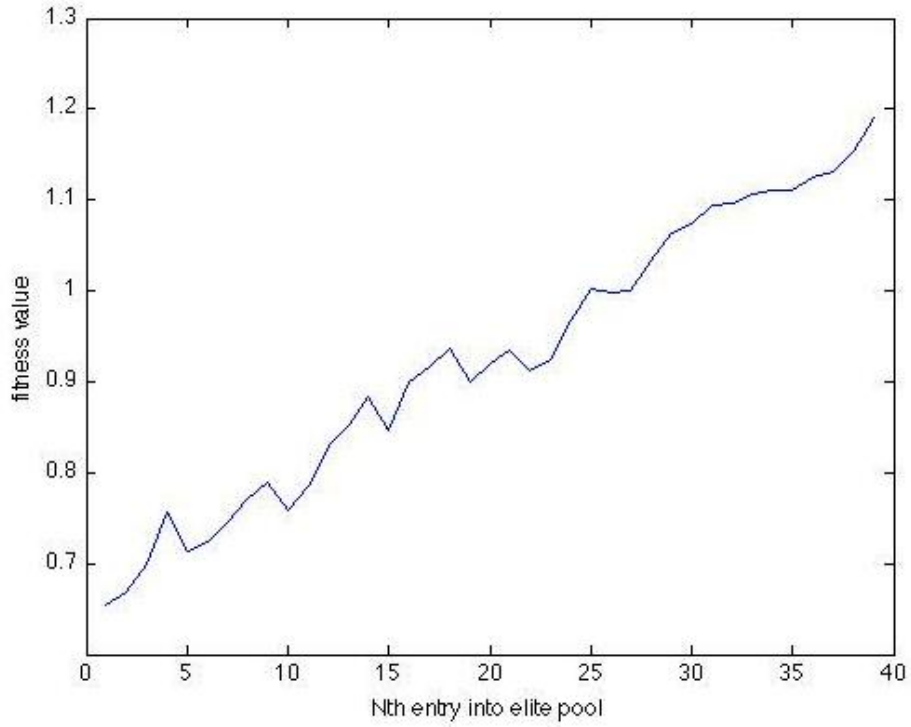


Figure 22. Maximization of I(X:Y) (flow variables – positive real numbers)

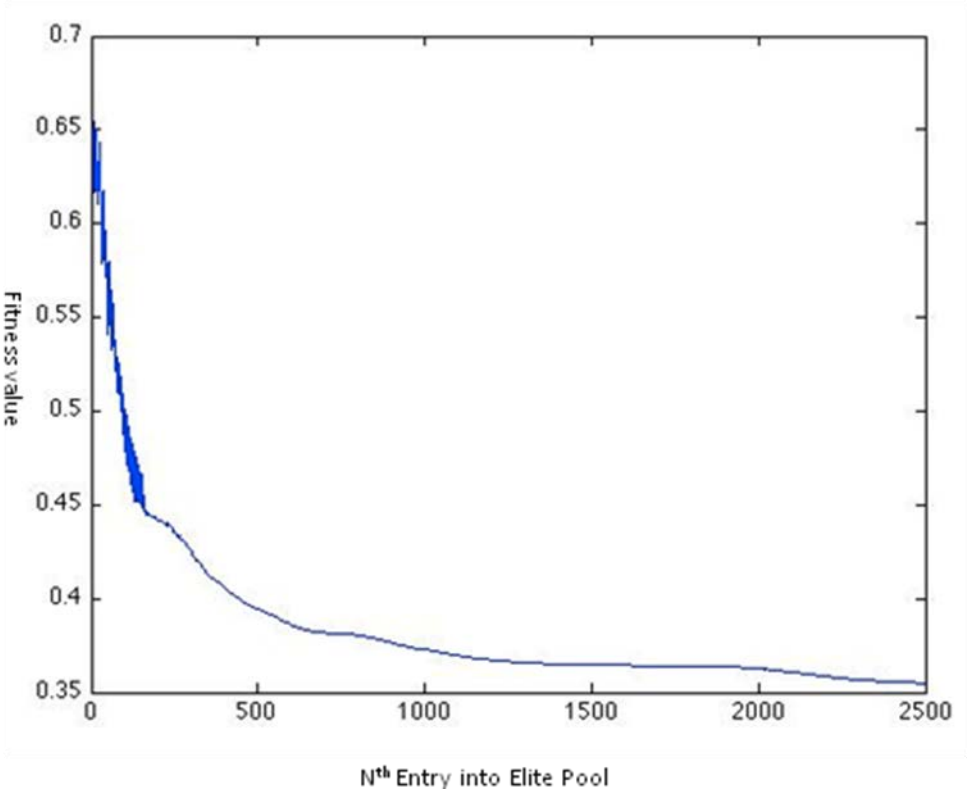


Figure 23. Minimization of I(X:Y) (flow variables – positive real numbers)

9.0 CONCLUSIONS

In this work, a systematic approach was developed to study the efficiency of networks. This approach uses graph theory to characterize the structure of the network and information theory to study its operation. A genetic algorithm was used to identify the minimum and maximum flow of the network. The ability to study and optimize network flow gives decision makers the ability to identify critical and vulnerable nodes and the opportunity to induce or mitigate congestion. Simple examples such as a logistics distributed network for Air Terminal Operations Flight (ATOF) were used to illustrate the approach. It was shown with specific numerical examples that the difference between the maximum and minimum information flows of a network can be quite significant. By identifying the maximum and minimum flows, a decision maker will have a better understanding of how to protect or attack a network system.

11.0 REFERENCES

[1] J. Cares, *Distributed Networked Operations- The Foundations of Network Centric Warfare*, Alidade Press, Newport, Rhode Island, 2005.
[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*, Prentice Hall, 1993.
[3] D. W. Repperger, R. L. Ewing, J. B. Lyons, K. A. Ferris "*Some Mathematical and Information-theoretic Viewpoints on Complex Networks*," Ohio 2008.
[4] L. T. Ford, Jr. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press (1962).
[5] T. Bäck, *Evolutionary Algorithms in Theory and Practice,* Oxford University Press, NY, 1996.
[6] J. H. Holland, "*Adaptation in Natural and Articial Systems*, The University of Michigan Press, Ann Arbor, MI, 1976.
[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
[8] K. A. De Jong, *An analysis of the behavior of a class of genetic adaptive systems*", PhD. Dissertation, Univ. of Michigan, Ann Arbor, MI, 1975.
[9] M. T. McMahon, *A distributed Genetic Algorithm with Migration for the Design of Composite Laminate Structures*. MS Thesis. Virginia Polytechnic Institute and State University, 1998.
[10] C. E. Shannon, "Communications in the Presence of Noise," *Proceedings of the IRE*, 37, pp. 10-22, 1949.
[11] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc. 1991.
[12] D. W. Repperger, et al., "Quantitative Measurements of System Complexity," USAF Patent application AFD 813, approved April, 2006, patent in progress.
[13] T. B. Sheridan and W. R. Ferrell, *Man-Machine Systems: Information, Control, and Decision Models of Human Performance*, The MIT press, Cambridge, Massachusetts, 1981.
[14] A. R. Sastry, "Autonomous Mobile Mesh Networks and Applications for Defense Network-centric Operations," *Proceedings of the SPIE*, 2006, Vol. 6249, 62490R-1-R-10.
[15] T. S. Bertulis and J. O. Miller, "Using Simulation to understand Interim Brigade Combat Team Munitions Logistics," *International Journal of Logistics: Research and Applications*, Vol. 8, No. 1, March, 2005, pp. 81-93.
[16] J. Seyba, J. Lyons, and D. Ames, "Computer-based Aerial Port Simulation (CAPS): Designing an Experiment Examining Team-based Logistics Collaboration," *Proceedings of the 2006 IEEE International Symposium on Collaborative Technologies and Systems*, May 14-17, 2006, pp. 226-231.
[17] J. Lyons, J. Ritter, K. Thomas, K. Militello, and P. Vincent, "Collaborative Logistics: Developing a Framework to Evaluate Socio-Technical Issues in Logistic-based Networks," *Proceedings of the 2006 IEEE International Symposium on Collaborative Technologies and Systems*, May 14-17, 2006, pp. 208-214.
[18] G. Zhou, H. Min, and M. Gen, "The Balanced Allocation of Customers to Multiple Distribution Centers in the Supply Chain Network: A Genetic Algorithms Approach," *Computers & Industrial Engineering*, 43 (2002), pp. 251-261.
[19] A. Syarif, Y-S Yun, and M. Gen, "Study on Multi-stage Logistic Chain Network: A Spanning Tree-based Genetic Algorithm Approach," *Computers & Industrial Engineering*, Vol. 43 (2002), pp. 299-314.
[20] H. Min, H. J. Ko, and C. S. Ko, "A Genetic Algorithms Approach to Developing the Multi-echelon Reverse Logistics Network for Product Returns," *Omega- The International Journal of Management Science*, Vol. 34 (2006), pp. 56-69.
[21] M. Braglia and R. Gabbrielli, "A Genetic Approach for Setting Parameters of Reorder Point Systems," *International Journal of Logistics: Research and Applications*, Vol. 4, No. 3, 2001, pp. 345-358.9
[22] W-C Chiang and R. A. Russell, "Integrating Purchasing and Routing in a Propane Gas Supply Chain," *European Journal of Operational Research*, Vol. 154 (2004), pp. 710-729.

[23] A. Rizzi, "Optimisation of Distributive Logistics Flows in Multidivisional Firms: The Case of the Ceramic Tiles Industry," *International Journal of Logistics: Research and Applications*, Vol. 3, No. 1, 2000, pp. 67-82.
[24] D. K. Sharma, D. Ghosh, and D. M. Mattison, "An Application of Goal Programming with Penalty Functions to Transhipment Problems," *International Journal of Logistics: Research and Applications*, Vol. 6, No. 3, 2003, pp. 125-136.
[25] J. Faulin, "Combining Linear Programming and Heuristics to Solve a Transportation Problem for a Canning Company in Spain," *International Journal of Logistics: Research and Applications*, Vol. 6, No. 1-2, 2003, pp. 17-27.
[26] G. N. Stock, N. P. Greis, and J. D. Kasarda, "Enterprise Logistics and Supply Chain Structure: The Role of Fit," *Journal of Operations Management*, 18, (2000), pp. 531-547.
[27] W-C Yeh, "A Hybrid Heuristic Algorithm for the Multistage Supply Chain Network Problem," *Int. J. Adv. Manuf. Technology* (2005), Vol. 26, pp. 675-685.
[28] R. S. Gaonkar and N. Viswanadham, "Strategic Sourcing and Collaborative Planning in Internet-enabled Supply Chain Networks Producing Multigeneration Products," *IEEE Transactions on Automation Science and Engineering*, vol. 2, No. 1, January, 2005, pp. 54-66.
[29] B. C. Rajeshkumar and T. RameshBabu, "Evaluation of Logistics Related Policies Between Two Different Levels of the Supply Chain Network- A Case Study," *Ann. Oper. Res.* (2006) 143, pp. 77-89.
[30] M. Ulieru and M. Cobzaru, "Building Holonic Supply Chain Management Systems: An e-Logistics Application for the Telephone Manufacturing Industry," *IEEE Transactions on Industrial Informatics*, Vol. 1, No. 1, February, 2005, pp. 18-30.
[31] J. Gjerdrum, N. J. Samsatla, N. Shah, and L. G. Papageorgiou, "Optimisation of Policy Parameters in Supply Chain Applications," *International Journal of Logistics: Research and Applications*, Vol. 8, No. 1, March 2005, pp. 15-36.
[32] M. Karkkainen, T. Ala-Risku, K. Framling, "The Product Centric Approach: A Solution to Supply Network Information Management Problems?," *Computers in Industry*, 52 (2003), pp. 147-159.
[33] M. Jahre and N. Fabbe-Costes, "Adaptation and Adaptability in Logistics Networks," *International Journal of Logistics: Research and Applications*, Vol. 8, No. 2, June, 2005, pp. 143-157.
[34] C. E. Riddalls, B. Icasati-Johanson, C. M. Axtell, S. Bennett, and C. Clegg, "Quantifying the Effects of Trust in Supply Chains During Promotional Periods," *International Journal of Logistics Research and Applications*, Vol. 5, No. 3, 2002, pp. 257-274.
[35] H. Kerivin, D. Nace, and T-T-L Pham, "Design of Capacitated Survivable Networks with a Single Facility," *IEEE/ACM Transactions on Networking*, Vol. 13, No. 2, April, 2005, pp. 248-261.
[36] D. W. Repperger and L. Rothrock, "Using GA-based Intelligent Control Means to Enhance Human-Machine Interfaces," Auto-*Soft – Intelligent Automation and Soft Computing*, Vol. 11, No. 2, 2005, pp. 123-140.
[37] D. W. Repperger, "Evolutionary Algorithms with Application to Biomedical Engineering," *Encyclopedia of Biomedical Engineering*, Wiley, Vol. 3, April, 2006, pp. 1519-1528.

APPENDIX
Matlab Code for Genetic Algorithm

I. —Assignfitness.m" function calculates the fitness value of a chromosome depending on which we manage the chromosomes in the elite pool.

Assignfitness.m

```
function res=Assignfitness(bits)

% This function calculates the fitness value (Information flowing in the
% network) depends on the chromosome.

buffer=Parsebits(bits);

sum=0;

for i=1:15

sum=sum+buffer(i);

end

sum=2*(sum)+7;

% Populating the Contingency Matrix

m(1,1)=(7.0+buffer(1)+buffer(2)+buffer(14)+buffer(4))/sum;

m(1,2)=(buffer(12))/sum;

m(1,3)=(buffer(13))/sum;

m(1,4)=(buffer(3))/sum;

m(1,5)=(buffer(15))/sum;

m(2,1)=(buffer(1))/sum;

m(2,2)=(buffer(12)+buffer(7))/sum;

m(2,3)=(buffer(8))/sum;

m(2,4)=0;

m(2,5)=(buffer(5))/sum;

m(3,1)=(buffer(2))/sum;

m(3,2)=(buffer(7))/sum;

m(3,3)=(buffer(6)+buffer(8)+buffer(13))/sum;
```

```
m(3,4)=(buffer(10))/sum;

m(3,5)=(buffer(9))/sum;

m(4,1)=(buffer(14))/sum;

m(4,2)=0;

m(4,3)=0;

m(4,4)=(buffer(3)+buffer(10))/sum;

m(4,5)=(buffer(11))/sum;

m(5,1)=(buffer(4))/sum;

m(5,2)=0;

m(5,3)=(buffer(6))/sum;

m(5,4)=0;

m(5,5)=(buffer(9)+buffer(15)+buffer(5)+buffer(11))/sum;

for a=1:5
    for b=1:5
        if m(a,b)==0

            m(a,b)=1;
        end;
    end;
end;

% Sum of the elements across the rows

r1=(7+buffer(1)+buffer(2)+buffer(14)+buffer(4)+buffer(12)+buffer(13)+buffer(3
)+buffer(15))/sum;

r2=(buffer(1)+buffer(12)+buffer(7)+buffer(8)+buffer(5))/sum;

r3=(buffer(2)+buffer(7)+buffer(6)+buffer(13)+buffer(8)+buffer(10)+buffer(9))/
sum;

r4=(buffer(14)+buffer(3)+buffer(10)+buffer(11))/sum;

r5=(buffer(4)+buffer(6)+buffer(15)+buffer(5)+buffer(9)+buffer(11))/sum;

% Sum of the elements across the Columns

c1=(7+buffer(1)+buffer(2)+buffer(14)+buffer(4)+buffer(1)+buffer(2)+buffer(14)
+buffer(4))/sum;
```

```
c2=(buffer(12)+buffer(12)+buffer(7)+buffer(7))/sum;

c3=(buffer(13)+buffer(6)+buffer(13)+buffer(6)+buffer(8)+buffer(8))/sum;

c4=(buffer(3)+buffer(3)+buffer(10)+buffer(10))/sum;

c5=(buffer(15)+buffer(5)+buffer(15)+buffer(5)+buffer(9)+buffer(11)+buffer(9)+
buffer(11))/sum;

% Calculating H(X) and H(Y)

Hx=(-((r1*log (r1))+(r2*log (r2))+(r3*log
(r3))+(r4*log(r4))+(r5*log(r5))))/(log (2));

Hy=(-((c1*log (c1))+(c2*log (c2))+(c3*log (c3))+(c4*log (c4))+(c5*log
(c5))))/(log (2));

Hxy=0;

for a=1:5

    for b=1:5

        Hxy=Hxy-(m(a,b)*log(m(a,b)));

    end;
end;

 Hxy=Hxy/log(2);

res=Hx+Hy-Hxy;
H1=Hx-res;
H2=Hy-res;
%res=H1+H2;
end
```

II. ~~–~~Gene_Algo m" performs the Genetic Algorithm required to produce the flow variable set that results in maximum information flow and minimum information flow.

Gene_Algo.m

```
Population=cell(20,2); % Create a 20 x 2 cell

E=cell2struct(Population,{'bits','fitness'},2); %converts a cell into struct

v=[1;2;2;3;1;3;1;2;1;3;1;3;2;4;3]; % initial chromosome in the customized
order.

c=dec12bin(v); % Converting the integer vector to binary string

for i=1:20
E(i).bits=c; % initialize the pool
```

```
end;

bfound=0;

while  bfound ~= 1

    Totalfitness=0;
    for i=1:20

        E(i).fitness=Assignfitness(E(i).bits);
        Totalfitness=Totalfitness+E(i).fitness;

    end;

    cPop=0; % Keeps track of number of successful iterations
    clear xl;

    while  cPop<150 % this number the number of successful iterations
required

        if cPop>0
            Totalfitness=0;

            % Calculating the total fitness value

            for i=1:20
                Totalfitness=Totalfitness+E(i).fitness;
            end;
        end;

        %offspring1 is selected randomly from the elite pool using Roulette
Function

        offspring1=Roulette(Totalfitness,E);
        offspring2=Roulette(Totalfitness,E);

        %Crossover is performed between offspring1 and offspring2

        [st1,st2]=Crossover(offspring1,offspring2);
        offspring1=st1;
        offspring2=st2;

        % offsprings are mutated

        offspring1=Mutation(offspring1);
        offspring2=Mutation(offspring2);

        % Checks whether this chromosome satisfies all the flow equations

        p1=Check(offspring1);

        if  p1>0
            af=Assignfitness(offspring1);   % Calculates the fitness value
```

```
             % sorts the Elite Pool
            % In the case of Maximization 1st element contains the maximum
            % value
            % In the case of Minimization 1st element contains the minimum
            % value

            [E,st]=sort_ga(E,af,offspring1); % Sorts the elite pool

            if st == 1
                cPop=cPop+1;
                xl(cPop,1)=cPop;
                xl(cPop,2)=af;
                %disp(af);
            end;
        end;

        % Same operations are performed for the second chromosome as well

        p2=Check(offspring2);
        if p2 > 0
            af=Assignfitness(offspring2);

            % sorts the Elite Pool
            % In the case of Maximization 1st element contains the maximum
            % value
            % In the case of Minimization 1st element contains the minimum
            % value

            [E,st]= sort_ga(E,af,offspring2);


            if st == 1
                cPop=cPop+1;
                xl(cPop,1)=cPop;
                xl(cPop,2)=af;
                %disp(af);
            end;

       end;

    end;% while cPop end

 bfound=1;

end; % while bfound end

% Saving and Plotting the fitness values.

xlswrite('data.xls',xl); % Saves the file as data.xls before execution one
needs to either change name here or in the workspace
```

```
plot(xl(:,1),xl(:,2));
xlabel('Number of Iterations');
ylabel('Fitness Value');
```

III. ~~r~~oulette.m" helps in finding out the chromosome for crossover

Roulette.m

```
function s= Roulette(total_fitness,E)
% This function selects the chromosome for crossover

    l=rand(1);
    Slice = l* total_fitness;
    FitnessSoFar = 0.0;
    k=0.0;
    for i=1:20
        k=k+E(i).fitness;
        FitnessSoFar = FitnessSoFar+E(i).fitness;
        if FitnessSoFar >= Slice
            s=E(i).bits;
            break;
        else
            s='';
       end;
    end;
 end
```

IV. ~~c~~rossover m" takes both the off springs as input and crossover them to produce new off springs

Crossover.m

```
function [s1,s2]=Crossover(s1,s2)

% Performs crossover between two chromosomes

s=rand(1);% generates a random number if the number is less than 0.6
multiplies that with 33 and
          % crossovers both the offsrpings from that point

if s<0.6   % 0.6 is the crossover rate

    crossoverrate=s * 33;
    l= round(crossoverrate);    % rounds the number to nearest integer
    tempstr=s2;

    if l>0

        for i=l:33
```

```
            s2(i)=s1(i);
            s1(i)=tempstr(i);

        end;

    end;

end;

end
```

V. —mutation m" performs the mutation operation on the offspring.

Mutation.m

```
function bits=Mutation(s)

%Mutate bits of the chromosome depending on the mutation rate

for i=1:33

    if rand(1)<0.001 % Mutation Rate

        j=s(i);

        if j=='1'

            s(i)='0';
        end;

        if j=='0'

            s(i)='1';
        end;

    end;


end;


buffer=Parsebits(s);

% Calculating the 4 depending variables

buffer(12)=buffer(1)+buffer(5)-buffer(7)+buffer(8);
buffer(13)=buffer(2)-buffer(6)+buffer(7)-buffer(8)+buffer(9)+buffer(10);
buffer(14)=buffer(3)+buffer(10)-buffer(11);
buffer(15)=buffer(4)-buffer(5)+buffer(6)-buffer(9)-buffer(11);

if buffer(12)>0 && buffer(13)>0 && buffer(14)>0 && buffer(15)>0
```

```
bits=dec12bin(buffer);

else

    bits='';

end;


end
```

VI. –sort_ga m" inserts the new chromosome in the elite pool based on its fitness value i.e.,while finding the maximum optimized information flow, the chromosome with least fitness value is discarded and new chromosome is added to the elite pool and elite pool is sorted accordingly.

Sort_ga.m

```
function [E,st]= sort_ga(E,s,bits)

% This function sorts the Elite Pool depending on the type of Optimization

    st=1;

 % Checks if this chromosome is already present or not

    for i=1:20
        if E(i).fitness==s
        st=0;
        break;
        end;
    end;

  % Checks whether the chromosome is bad when compared to worst chromosme
  % in the Elite Pool

    if s < E(20).fitness
        st=0;
    end

    if  st ~= 0
        for i=1:20

            % '<' is for maximization and '>' is for minimaization in the
            % below statement
            % For minimization it is " if E(i).fitness > s "
            % For maximization it is " if E(i).fitness < s "

            if E(i).fitness < s
                for k= 19:-1:i
                    E(k+1).bits=E(k).bits;
                    E(k+1).fitness=E(k).fitness;
                end;
```

```
                E(i).bits=bits;
                E(i).fitness=s;
                st=1;
                break;
            end;
        end;
    end;
end
```

VII. –dec12bin.m" is used to convert the chromosome from decimal format to binary format so that it would be easy for crossover and mutation.

Dec12bin.m

```
function bin=dec12bin(k)
% This function converts the chrosomes from integer to binary format so that
it would be
%easy for crossover and mutation
bin='';
    for i=1:15
        num=k(i,1);
        g= dec2bin(num);
        len=length(g);
        if len<3

            for q=1:3-len
                g=strcat('0',g);
            end;
        end;
        bin=strcat(bin,g);
    end;
end
```

VIII. –parsebits m" converts the chromosome from binary to interger

Parsebits.m

```
function x=Parsebits(s)
% This function cpnverts the chromosome from binary format to decimal format
%so that finding the fitness value would be easy

    for i=1:45
        if rem(i,3)==1
            tem=0;
            g=bin2dec(s(i));
            d=g*4;
            tem=d;
        end

        if rem(i,3)==2
            g=bin2dec(s(i));
            d=g*2;
            tem=tem+(d);
        end
```

```
        if rem(i,3)==0
            g=bin2dec(s(i));
            tem=tem+g;
            v(i/3)=tem;

        end

    end
x=v';
end
```

IX. –eheck m" checks whether the given chromosome satisfies the flow equations or not.

Check.m

```
function s = Check(v)

% this function checks whether the chromosome length is 45 or not

    s=1;

    if length(v)>45 || length(v)<45
        s=0;
    else

        s=1;
    end

end
```

X. –Generate.m" is used to generate the candidate solution for the genetic algorithm and the code that does this is given below.

Generate.m

```
function buffer=generate(xcurrent,p,N)

% Takes the current solution and search in its vicinity to find the next
solution

sum=0;
for i=1:11
    z(1,i)=1;
end

for i=1:11
    p(1,i)= p(1,i)-(2.5*z(1,i));
end

for h=1:11
    sum=sum+(p(1,h)*N(:,h));
end
```

```
% Calculating Another solution with in the vicinity of the current solution

buffer=xcurrent+sum;

%Calculates the remaining 4 dependent variables

buffer(12)=buffer(1)+buffer(5)-buffer(7)+buffer(8);
buffer(13)=buffer(2)-buffer(6)+buffer(7)-buffer(8)+buffer(9)+buffer(10);
buffer(14)=buffer(3)+buffer(10)-buffer(11);
buffer(15)=buffer(4)-buffer(5)+buffer(6)-buffer(9)-buffer(11);


end
```

XI. —Mod_ga m" performs the genetic algorithm using the chromosomes that were produced using above generate method. It is similar to the —gne_algo.m" except that the chromosomes are generated in a different way.

Mod_ga.m

```
Population=cell(20,2);  % Create a 20 x 2 cell

E=cell2struct(Population,{'bits','fitness'},2); %converts a cell into struct

% Co-efficient Matrix

A=[1,1,-1,1,0,0,0,0,0,0,0,-1,-1,1,-1;
    -1,0,0,0,-1,0,1,-1,0,0,0,1,0,0,0;
    0,0,0,-1,1,-1,0,0,1,0,1,0,0,0,1;
    0,-1,0,0,0,1,-1,1,-1,-1,0,0,1,0,0;
    0,0,1,0,0,0,0,0,0,1,-1,0,0,-1,0];


[b c d]=svd(A);

% Calculating the Null space of A

N=d(:,5:15);

% initial chromosome in the customized order.

v=[1;2;2;3;1;3;1;2;1;3;1;3;2;4;3];

c=dec12bin(v); % Converting the integer vector to binary string

xcurrent1=v;
xcurrent2=v;

% initialize the pool

for i=1:20
E(i).bits=c;
end;
```

```
bfound=0;

while  bfound ~= 1

    Totalfitness=0;

    for i=1:20

        E(i).fitness=mod_Assignfitness(v);
        Totalfitness=Totalfitness+E(i).fitness;

    end;

    cPop=1;     % Keeps track of number of successful iterations

    clear xl;

    xl(cPop,1)=cPop;
    xl(cPop,2)=E(1).fitness;

    while  cPop<5  % this number the number of successful iterations required

         %offspring1 is selected randomly from the elite pool using
         %Roulette Function

        offspring1=Roulette(Totalfitness,E);
        offspring2=Roulette(Totalfitness,E);

        %Crossover is performed between offspring1 and offspring2

        [st1,st2]=Crossover(offspring1,offspring2);
        offspring1=st1;
        offspring2=st2;

         % offsprings are mutated

        offspring1=mod_Mutation(offspring1);
        offspring2=mod_Mutation(offspring2);

        % The binary Chromosome is converted into integer vector.

        p1=Parsebits(offspring1);
        p2=Parsebits(offspring2);

        % Generates the next solution by searching in the vicinity of
        % current solution

        xcandidate1=generate(xcurrent1,p1',N);
        xcandidate2=generate(xcurrent2,p2',N);

        % Checks whether this chromosome satisfies all the flow equations
```

```
            q1=mod_Check(xcandidate1);

            if norm(q1)>0

                af=mod_Assignfitness(xcandidate1);% Calculates the fitness
value

                 % sorts the Elite Pool
            % In the case of Maximization 1st element contains the maximum
            % value
            % In the case of Minimization 1st element contains the minimum
            % value

            [E,st]=sort_ga(E,af,offspring1); % Sorts the elite pool

                if st == 1
                    cPop=cPop+1;
                    xl(cPop,1)=cPop;
                    xl(cPop,2)=af;
                     %disp(af);
               end;

            end;



        % Same operations are performed for the second chromosome as well

            q2=mod_Check(xcandidate2);

            if norm(q2)>0

                af=mod_Assignfitness(xcandidate2);
                  % sorts the Elite Pool
            % In the case of Maximization 1st element contains the maximum
            % value
            % In the case of Minimization 1st element contains the minimum
            % value

            [E,st]=sort_ga(E,af,offspring1); % Sorts the elite pool

                if st == 1
                    cPop=cPop+1;
                    xl(cPop,1)=cPop;
                    xl(cPop,2)=af;
                     %disp(af);
               end;

            end;
```

```
    end;% while cPop end

bfound=1;

end;% while bfound end

% Saving and Plotting the fitness values.

xlswrite('data.xls',xl); % Saves the file as data.xls before execution one
needs to either change name here or in the workspace

plot(xl(:,1),xl(:,2));
xlabel('Number of Iterations');
ylabel('Fitness Value');
```

User Manual:

This section explains how to run the code that is presented above.

The assumption is that Matlab is already installed and have all the matlab files in the workspace.

To run the algorithm for integer values, the user would run _Gene_Algo.m‘. This is done by typing _Gene_Algo‘ in the Matlab editor. The results are plotted and are saved in the worksapce in the form of excel file. The only difference in the maximization and minization is in _sort_ga.m‘function, which is well documented in the function file.

To run the algorithm for real values, the user would run _mod_GA.m‘. This is performed in the same way as _Gene_Algo.m‘. Some of the functions differ from integer algorithm. The functions required for this will be saved in the folder –modified” in the CD that is submitted.

LIST OF ACRONYMS, ABBREVIATIONS, AND SYMBOLS

Acronyms:

ATOF	Air Terminal Operation Flight
CAPS	Computer-based Aerial Port Simulation
CS	Cargo Services
FS	Fleet Services
GA	Genetic algorithm
PS	Passenger Services
RS	Ramp Services
sGA	Standard Genetic algorithm

Symbols:

H(X)	Input Uncertainty
H(Y)	Output Uncertainty
H(X/Y)	Equivocation
H(Y/X)	Spurious Uncertainty
MI	Mutual Information (measure)
I(X;Y)	Mutual Information (measure)
E_f	Efficiency Normalization (metric)
D_R	Information Distance